UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Grado en Ingeniería del Software

Universidad Politécnica de Madrid

ETSI SISTEMAS INFORMÁTICOS

PROYECTO DE FIN DE GRADO

Desarrollo de un sistema de gestión de tutorías a través de una aplicación móvil

Autor

Alberto Guerrero Martín

Tutor

Pedro Pablo Alarcón Cavero

Agradecimientos

A mi tutor Pedro Pablo y a Agustín, por toda su ayuda en el desarrollo de este trabajo y su memoria.

A la Delegación de Alumnos de la UPM, por hacer cada día un poco mejor esta universidad.

A mi familia y amigos, por apoyarme a lo largo de estos 4 años.

Abstract

Con la irrupción y el auge de las tecnologías móviles en estos últimos años, se ha hecho patente que los procesos de gestión tienden a ser controlados a través de estas tecnologías, permitiendo al usuario centralizar todos los servicios que le sea posible en un dispositivo de uso tan común como el teléfono móvil, así como acceder a ellos de forma rápida y cómoda.

El sistema de tutorías de la Universidad Politécnica de Madrid no es la excepción. Desde su creación, el Grupo de Innovación Educativa Tutorial Action (GIETA) [1] ha trabajado en la modernización del proceso de gestión que conlleva el sistema de tutorías, buscando las deficiencias del sistema tradicional que pudiesen resolverse utilizando la tecnología. Este Trabajo de Fin de Grado (TFG) ha tenido como objetivo apoyar la labor iniciada por el GIETA, desarrollando un sistema de gestión de tutorías mediante una aplicación móvil.

Para lograr este objetivo, inicialmente se realizó un proceso de reflexión sobre, entre otras cuestiones, la razón de ser de la aplicación a desarrollar, las ventajas que aportaría al usuario final, aquellos riesgos que podían amenazar al proyecto, etc. Este proceso se englobó dentro de la herramienta *Agile Inception Deck*.

Tras este proceso de reflexión, se estructuró el proyecto en fases o sprints de desarrollo, en las que se llevó a cabo la implementación del resultado final de este TFG, una aplicación móvil para el sistema operativo Android, que aporta funcionalidad que resuelve todos los requisitos asociados a las distintas historias de usuario definidas para el proyecto.

—

With the rise of mobile technologies in recent years, it has become clear that management processes tend to be controlled through these technologies, allowing users to centralize all services as possible, using a device as common as the mobile phone, and access them quickly and easily.

The tutorial system at the Technical University of Madrid is not an exception. Since it's creation, the Group of Educational Innovation Tutorial Action has worked on the modernization of the management process that involves the tutorial system, looking for weaknesses of the traditional system that could be solved using technology. This TFG has aimed to support the work initiated by the GIETA, developing a tutorship management system through a mobile application.

To achieve this goal, initially a process of reflection was held about, inter alia, the rationale for the application to be developed, the advantages it would bring to the final user, the risks that could threaten the project, etc. This process is encompassed within the *Agile Inception Deck* tool.

After this process of reflection, the project was divided into phases or sprints, in which took place the implementation of the outcome of the TFG, a mobile application for the Android operating system, which provides functionality that meets all the requirements associated with the different user stories defined for the project.

Índice

Índice de figuras

1- Introducción

Este Trabajo de Fin de Grado (TFG) surge de la necesidad, detectada por el consolidado Grupo de Innovación Educativa Tutorial Action (GIETA) [1] de la Universidad Politécnica de Madrid, de proporcionar a docentes y alumnos una herramienta que les permita hacer uso del sistema de tutorías de la UPM, de una manera mas integrada con las tecnologías actuales.

El objetivo principal que se pretende conseguir a lo largo de este TFG es el desarrollo de un sistema de gestión de tutorías a través de una aplicación móvil. Con este objetivo, este trabajo pretende resolver algunos de los principales problemas observados por el GIETA en el sistema tradicional de tutorías, así como acercar mas el uso de las tutorías tanto a alumnos como a profesores.

Asimismo, con el desarrollo de esta aplicación se pretende conseguir una herramienta de gestión mas acorde con la sociedad actual, que se encuentra rodeada de tecnología móvil. Una herramienta que facilite al usuario el uso del sistema de tutorías, y que mejore la calidad del mismo, resolviendo los problemas presentes en el sistema tradicional.

Con independencia de las ventajas que pretende aportar este TFG a los usuarios finales de las tutorías, existe un fuerte componente motivacional a la hora de proponer este proyecto, componente basado en la posibilidad de desarrollar un sistema que siga la arquitectura cliente-servidor, y la implemente completamente, desde la base de datos y la aplicación de lado servidor o backend, hasta la aplicación cliente o frontend. Desarrollar un sistema de esta considerable magnitud permite entender mejor el funcionamiento de este tipo de arquitectura, así como aportar experiencia y conocimiento sobre las últimas tecnologías que permiten implementar los componentes del sistema.

En esta memoria se plasma el trabajo llevado a cabo a lo largo de este TFG, documentándose en detalle el proceso seguido, tanto la fase inicial de introspección sobre el propio proyecto, como la fase de implementación técnica del mismo.

2- Descripción del problema

Desde su creación, el Grupo de Innovación Educativa Tutorial Action (GIETA), y dentro de él el Proyecto de Innovación Educativa Tutorial Action (PIETA) [2] ha trabajado estudiando el funcionamiento actual de las tutorías en las asignaturas impartidas en la Universidad Politécnica de Madrid, con el objetivo principal de descubrir las deficiencias que el sistema actual pueda tener, y proponer mejoras que las suplan.

Tras este estudio, cuyos detalles pueden consultarse en artículos como el publicado en la revista de docencia universitaria REDU [3], el proyecto detectó los siguientes problemas:

- **Gran abandono del uso del sistema**: un gran porcentaje del alumnado (la gran mayoría) no hacía uso de las horas de tutorías de las que dispone cada profesor, reservando sus dudas para las horas de docencia (que no pueden dedicarse a responder preguntas complejas) o incluso para sus propios compañeros antes que para el profesor.
- **Overbooking de las tutorías en fechas críticas**: paradójicamente con lo expuesto en el punto anterior, en ciertas asignaturas se observó que el uso de las tutorías aumentaba en fechas concretas, normalmente coincidentes con la proximidad de una prueba escrita de la asignatura, o con la entrega de alguna actividad práctica.
- **Incertidumbre en cuanto a la disponibilidad del profesor**: al tener como única información el horario de tutorías de cada profesor, comúnmente los alumnos que acuden a tutorías encuentran al profesor ocupado atendiendo a otros alumnos, lo cual satura ciertas horas de tutorías con mas demanda, e impide que se desarrollen de la forma mas ideal posible y dedicando el tiempo necesario a cada alumno. Así mismo, la desinformación es prácticamente total ante cualquier imprevisto, por ejemplo indisposición del profesor, lo cual puede generar malestar y descontento con el sistema de tutorías entre los alumnos afectados.

En este contexto, el GIETA lleva años realizando avances que permitan resolver estos problemas, entre ellos la aplicación web Tutorial Action [4], que supuso un primer paso en la modernización del uso del sistema de tutorías. Con este TFG se busca profundizar mas en estos avances, particularmente en las ventajas que aportan las tecnologías móviles mencionadas en el punto anterior, desarrollando un sistema de gestión móvil de las tutorías.

3- Metodología de desarrollo

En las primeras reuniones del proyecto, que tuvieron como objetivo sentar sus bases, se decidió que la metodología mas adecuada para guiar el desarrollo del proyecto pasaba por ser una **metodología ágil** de desarrollo. Particularmente se optó por la metodología Scrum [5], por la experiencia previa tanto del alumno (único desarrollador del proyecto) como del tutor utilizando esta metodología. A lo largo de esta sección, se detallan las características de Scrum, así como todas las particularidades relacionadas con la gestión de este proyecto.

3.1- Conceptualización

De forma previa a la fase de desarrollo del sistema, es necesario llevar a cabo un proceso de conceptualización sobre el proyecto propuesto, en el que se cumplan los siguientes objetivos:

- Visión global del proyecto: uno de los objetivos básicos de este estudio previo del proyecto debe ser aportar una visión en conjunto de todo el sistema a desarrollar, que permita tanto al equipo como al cliente tener claros los objetivos con respecto al proyecto, algo clave a la hora de estimar y planificar.
- Planificación del desarrollo: otro de los objetivos vitales de esta fase es estructurar el proceso de desarrollo del proyecto, realizando el reparto de la carga de trabajo, estimando la duración total del desarrollo, la cantidad de recursos que serán necesarios, y otras tareas similares.
- Viabilidad y riesgos: en esta fase es imprescindible estudiar la viabilidad del proyecto, que no debe comenzarse a desarrollar en caso de que no se considere viable. Para llevar a cabo dicho estudio, es necesario conocer los riesgos a los que está expuesto nuestro proyecto, y en que medida.

Como herramienta para llevar a cabo este proceso de conceptualización, en este TFG se ha elegido el *Inception Deck*, con la descripción y estructuras especificadas en el libro *"The Agile Samurai"* [6] de Jonathan Rasmusson.

El *Inception Deck* es una herramienta pensada para servir de soporte inicial a la gestión de proyectos mediante metodologías ágiles de desarrollo, que está compuesto de una serie de tareas que el equipo debe llevar a cabo antes de abordar cualquier proyecto, de las cuales se ha eliminado la última de ellas por no considerarla acorde al perfil del proyecto.
Tal y como Rasmusson explica en su libro, esta herramienta está pensada para ayudar a los equipos a hacerse una serie de preguntas clave antes de empezar cualquier proyecto, de cara a estudiar su viabilidad, costes, planificación, etc.

Tras las modificaciones realizadas al modelo de *Inception Deck* propuesto por Rasmusson, quedan finalmente las siguientes etapas o tareas:

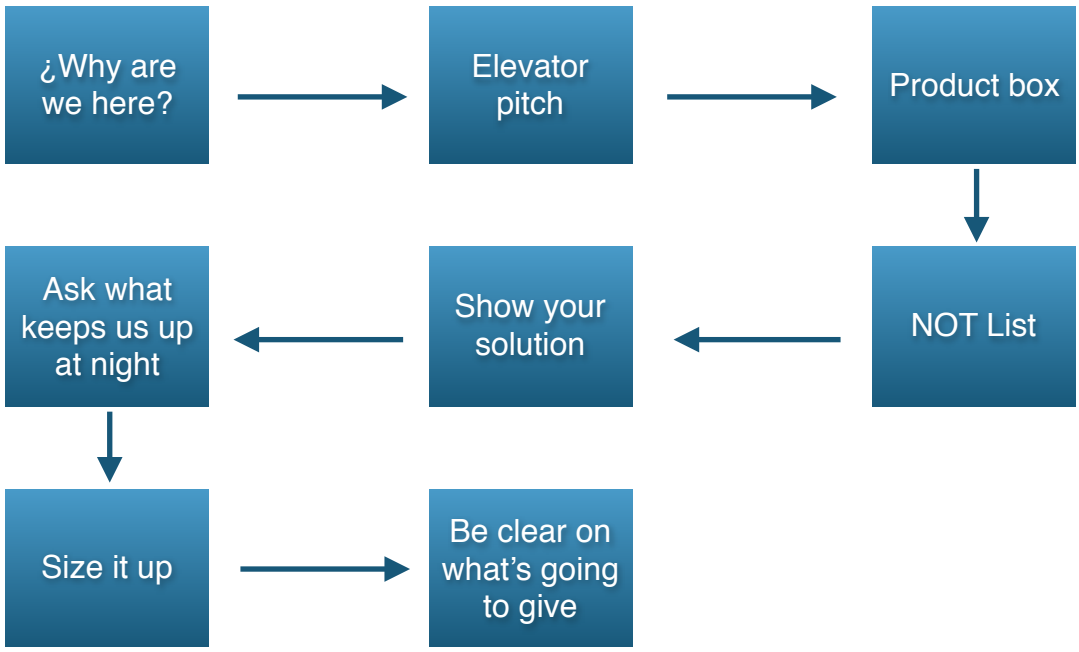

Figura 1: Gráfico de tareas que componen el Inception Deck, basado en [6]

A continuación se resumen los objetivos principales de cada una de las tareas del I*nception Deck*:

1. **Why are we here?**: tiene como objetivo principal recordar al equipo el por qué del proyecto, y a quién está dirigido.
2. **Elevator pitch**: su objetivo es obtener una breve descripción del proyecto, que sea rápida, clara y concisa.
3. **Product box**: imagen que, mediante metáforas visuales, resuma las capacidades de nuestro sistema de forma atractiva para el usuario, induciéndole a utilizarlo.
4. **NOT list**: listado con lo que se está y lo que **no** se está haciendo con este proyecto.
5. **Show your solution**: visualización esquemática a nivel técnico del sistema que se pretende desarrollar.
6. **Ask what keeps us up at night**: estudio de los riesgos del proyecto, así como de las medidas a tomar para evitarlos.
7. **Size it up**: proceso de construcción de la hoja de ruta del sistema.
8. **Be clear on what's going to give**: balanceo de aspectos del proyecto tales como tiempo, presupuesto, alcance, calidad, etc, para mantener la estabilidad del mismo.

La última de las herramientas de un *Inception Deck* al uso (*Show what it's going to take*) no ha sido finalmente utilizada en este trabajo. En este paso final, se conforma el equipo de desarrollo, poniendo especial énfasis en designar a aquel miembro del equipo que será el encargado de llevar las riendas del mismo, y de tomar las decisiones. Una vez configurado el equipo, se lleva a cabo una estimación del coste total que supondrá dicho equipo, calculando durante cuanto tiempo se necesitará a esas personas, y bajo que coste por unidad de tiempo.

Puesto que este trabajo no se corresponde con un proyecto de desarrollo al uso, no hay equipo que conformar, puesto que solo existe un único desarrollador (el alumno que realiza el trabajo), y tampoco supone ningún gasto, por lo que no es necesario realizar una estimación de gastos. En vista de lo expuesto, se ha retirado de este *Inception Deck* este paso final.

3.2- Sprints

Durante el Inception Deck se lleva a cabo, entre otras tareas, el diseño y estructuración del proceso de desarrollo. Siguiendo las directrices de la metodología Scrum, dicho proceso ha sido estructurado en **sprints**, pequeñas fases de desarrollo con un número concreto de características del sistema o historias de usuario a implementar, que en conjunto constituyen la totalidad del sistema.

Las ventajas de utilizar este sistema radican principalmente en la modularización del proceso de desarrollo. Al estructurar el periodo de desarrollo en estas fases de corta duración, la estimación de la duración total del proceso de desarrollo mejora considerablemente, ya que el margen de error al estimar tiempos es mayor si se estima el tiempo total del proyecto que si se estiman pequeños módulos del mismo.
Asimismo, la percepción del cliente mejora, ya que cada sprint genera como resultado una versión entregable del sistema con mayor funcionalidad que la anterior, lo que le da al cliente la seguridad de que el desarrollo de su producto está avanzando conforme tanto a los requisitos como a las historias de usuario. Además, en caso de modificarse el equipo de desarrollo, los retrasos por este cambio se verían minimizados, ya que el nuevo equipo podría partir del trabajo ya avanzado en los anteriores sprints.

Características de cada sprint

Un paso necesario antes de la definición de los sprints es determinar la duración de cada sprint, lo que ayudará a dividir mejor el proyecto en módulos cuyo desarrollo completo pueda asumirse en el tiempo de cada sprint. Para este proyecto, se ha definido la duración de cada sprint en **2 semanas**.

En lo referente al contenido de los sprints, cada uno constará de la siguiente información:

- **Sprint backlog**: fase previa a la descripción del sprint, en la que se recapitulan las historias de usuario que van a tratarse en dicho sprint.
- **Detalle del sprint**: desarrollo sobre el trabajo llevado a cabo en el sprint.
- **Modelo de datos**: estado en cada sprint del esquema representativo del modelo de datos de la aplicación. El objetivo de incluir este modelo en cada sprint es mantener la trazabilidad de los cambios que se produzcan en el esquema de datos del sistema.
- **Diagrama de clases**: diagrama representativo de las clases que componen cada una de las aplicaciones, así como las relaciones entre ellas. Al igual que con el modelo de datos, el objetivo que se persigue es trazar las modificaciones, en este caso las de las aplicaciones, que se produzcan en cada sprint.
- **Sprint review**: breve recapitulación de los avances realizados en el desarrollo del sistema durante el sprint. En esta sección se incluirá un *burndown chart* que represente como se han llevado a cabo los avances a lo largo del sprint, con respecto al tiempo.
- **Retrospectiva**: tabla resumen de acciones especialmente destacables relacionadas con el sprint (OK), acciones mejorables, y acciones claramente erróneas (KO).

3.3- Roles

Como ya se ha comentado, este proyecto no tiene un equipo de desarrollo al uso, ya que corresponde a un trabajo individual. Sin embargo, de cara a dar una mayor efectividad a los sprints, se ha considerado necesario realizar el establecimiento de roles de los stakeholders principales del proyecto: el profesor-tutor, y el alumno.

El reparto de roles queda de la siguiente manera:

- **Product owner**: persona responsable del producto en desarrollo, encargada de dar su visto bueno al trabajo realizado en cada sprint. En este caso, este rol lo asumirá el profesor-tutor de este trabajo, Pedro Pablo Alarcón Cavero.
- **Equipo de desarrollo**: grupo de desarrolladores que se encargarán de implementar el sistema. En este caso, el equipo estará compuesto por el único desarrollador asociado a este proyecto: el alumno que realiza el trabajo, Alberto Guerrero Martín, que asumirá los roles de desarrollador Android, desarrollador backend, administrador de la base de datos, y tester del sistema.

4- Desarrollo y gestión del proyecto

4.1- Inception Deck

4.1.1- Why are we here?

En el apartado 2 de esta memoria se han expuesto los principales problemas detectados por el GIETA en el sistema actual de tutorías. En esta primera etapa del *Inception Deck* se busca incidir en las tareas a realizar de cara a resolver estos problemas, y en la aportación de este TFG a ese objetivo.

Vistos los problemas, el GIETA trazó la siguiente línea de objetivos:

- **Aumentar el uso de las tutorías por parte de los estudiantes**: de cara a concienciar al alumnado de la utilidad de las tutorías, se observó que se requería un esfuerzo tanto en materia de difusión de la información referente a las mismas (especialmente los horarios de los profesores), como de la utilidad que pueden llegar a tener para los estudiantes con dudas, desde la ventaja obvia de la resolución de las mismas, hasta otro tipo de ventajas como, por ejemplo, una posible mejora de la calificación final de la asignatura al haber asistido a tutorías, puesto que no dejan de ser horas de trabajo del estudiante.
- **Facilitar la gestión**: uno de los puntos de trabajo básicos consistía en estudiar y proponer una modernización del sistema actual de gestión de las tutorías, desarrollando una herramienta que adapte el sistema de tutorías a la realidad social actual, muy marcada por el uso de tecnologías como la web o los móviles inteligentes. Entre otras ventajas, este nuevo sistema de gestión debe permitir:
  - *Conocer la disponibilidad del profesor*: conocer las franjas horarias para atender tutorías. Esto evita la incertidumbre de los alumnos a la hora de asistir a tutorías, puesto que sabrán en todo momento si el profesor está disponible para atenderles.
  - *Reservar tutorías*: de cara a evitar el overbooking anteriormente expuesto, el nuevo sistema deberá permitir la reserva de tutorías por parte de los alumnos. Esta funcionalidad aumenta la calidad de las tutorías, tanto para el alumno, que tiene mucha mas seguridad a la hora de asistir a una, como para el profesor, que puede planificar mejor las sesiones de tutorías, al tener constancia de qué alumnos van a asistir en cada momento.
  - *Indicar el motivo de la tutoría*: dando esta posibilidad se pretende ahondar aun más en la mejora de la calidad de las tutorías, permitiendo al profesor anticiparse a las dudas que los alumnos vayan a plantearle en las sesiones de tutorías, y por tanto llevar ya las respuestas preparadas, lo que permitirá optimizarlas en gran medida.

- *Sistema de notificaciones*: el sistema debe permitir notificar avisos, tanto a profesores (por ejemplo, una nueva reserva de tutoría que se les haya asignado) como a alumnos (por ejemplo, cancelación de una tutoría por algún tipo de imprevisto justificable).

Como resultado de esta línea de objetivos, nació la ya en funcionamiento aplicación web Tutorial Action, desarrollada por el grupo GIETA, buscando dar respuesta a los problemas detectados por el proyecto. Trasladando la gestión de las tutorías a una plataforma online, se consigue potenciar la difusión y el uso de esta herramienta por parte de los alumnos, que encuentran en la aplicación web un sistema mas moderno y completo que el tradicional, mas rudimentario.

A través de la aplicación, los alumnos pueden consultar la disponibilidad de sus profesores dentro de sus horarios de tutorías, evitando desplazarse hasta su despacho y encontrárselo atendiendo a otro alumno. Así mismo, pueden reservar la hora que mas les convenga para la tutoría. Por otro lado, al incluir el asunto de la tutoría en la reserva, el profesor puede llevarla preparada en gran parte.

Esta modernización del sistema de tutorías aumenta la calidad de esta herramienta, no obstante sigue teniendo vías de continuación. En el marco de dichas vías surge este TFG, consistente en el desarrollo de una aplicación móvil para la plataforma Android, y una API (*Application Programming Interface*) como soporte para la aplicación móvil, que en conjunto proporcionen una funcionalidad similar a la aplicación web, suponiendo un paso mas en la modernización del sistema, y un considerable avance en su nivel de usabilidad.

Con el desarrollo de esta aplicación móvil, se obtiene como resultado una interfaz con el alumno mas cómoda aun que la aplicación web, ya que no depende de un ordenador para utilizarse, tan solo de un móvil inteligente o Smartphone.

Dado que mas del 95% de los alumnos tienen un teléfono móvil de estas características, las aplicaciones llegarán a prácticamente la totalidad de los estudiantes, que podrán beneficiarse de la comodidad de llevar todo el sistema en el bolsillo, pero con la misma funcionalidad que la aplicación web, que podrá utilizarse a la par que las aplicaciones móvil, dando lugar a un sistema altamente flexible en cuanto al uso.

4.1.2- Elevator Pitch

El *elevator pitch* es una técnica que tiene como objetivo obtener una forma rápida de comunicar la esencia del un proyecto en un corto periodo de tiempo, y de una forma clara y concisa.

Un buen *elevator pitch* aportará las siguientes ventajas:

- **Claridad sobre el propósito del proyecto**: el elevator pitch responde a preguntas muy directas sobre que es el proyecto y a quien está dirigido. Esto puede observarse en la estructura descrita mas adelante.
- **Enfoca al equipo de desarrollo a pensar en el usuario**: debido a su estructura, el equipo de desarrollo se ve obligado a centrarse en qué hace el producto, y en por qué los usuarios lo preferirán frente a la competencia. Si bien es una herramienta pensada para equipos de desarrollo, su uso en este trabajo ha sido igualmente útil de cara a orientar el resultado final mucho mas al usuario.
- **Se centra en lo vital del proyecto**: proporciona al equipo claridad sobre el propósito del proyecto, lo cual ayuda a la hora de establecer prioridades, y de distinguir lo que realmente importa del proyecto.

El resultado que genera el *elevator pitch* es una frase, acorde a una estructura, que cumple los requisitos y proporciona las ventajas anteriormente descritas. Para este proyecto, se ha utilizado la plantilla propuesta por Jonathan Rasmusson en su libro *"The Agile Samurai"*, si bien no existe una forma única de hacer un *elevator pitch*.

La plantilla es la siguiente:

- **Para** [usuario objetivo]
- **que** [necesidad]
- **el** [nombre del producto]
- **es** [categoría del producto]
- **que** [beneficio clave del producto]
- **al contrario** [principal alternativa competitiva]
- **nuestro producto** [diferenciación principal del producto]

A continuación se describe mas en detalle la plantilla:

- **Para** [usuario objetivo]: explica a quien está dirigido el producto.
- **Que** [necesidad]: especifica el problema que el producto está destinado a resolver.
- **El** [nombre del producto]: el nombre de nuestro producto, que debe estar relacionado con el problema a resolver.
- **Es** [categoría del producto]: explica lo que es el producto.
- **Que** [beneficio clave del producto]: que tiene nuestro producto que hará que el usuario lo utilice en primer lugar.
- **Al contrario** [principal alternativa competitiva]: competencia actual, a la cual nuestro producto pretende mejorar.
- **Nuestro producto** [diferenciación principal del producto]: explica por qué nuestro producto es mejor que lo ya disponible.

Tras el uso de la plantilla descrita, el *elevator pitch* para este trabajo queda como sigue:

- **Para** los estudiantes y profesores de la UPM
- **que** hacen uso del sistema de tutorías de la Universidad
- **la** aplicación móvil Tutorial Action
- **es** un sistema de gestión de tutorías
- **que** permite controlarlas de forma telemática
- **al contrario** que el sistema tradicional
- **nuestro producto** permite gestionar las tutorías en cualquier lugar, en cualquier momento, y de forma cómoda.

“Para los estudiantes y profesores de la UPM que hacen uso del sistema de tutorías de la Universidad, la aplicación móvil Tutorial Action es un sistema de gestión de tutorías que permite controlarlas de forma telemática. Al contrario que el sistema tradicional, nuestro producto permite gestionar las tutorías en cualquier lugar, cualquier momento, y de forma cómoda.”

4.1.3- Product Box

Una vez se ha definido bien el producto, y los usuarios a los que está destinado, es interesante tener una herramienta de captación de usuarios del mismo, que sea a la vez vistosa y un buen resumen de las necesidades que cubre el producto.
Dentro del conjunto de herramientas que componen el *Inception Deck*, disponemos del *Product Box*, una herramienta pensada para generar una "caja de producto" para nuestro software, es decir, el diseño de un anuncio para nuestro producto, pensado como si fuese a colocarse en una caja (aunque nuestro producto sea un producto software que no se distribuye físicamente en cajas).

Para el diseño del *Product Box* correspondiente al producto software desarrollado en este TFG, se han seguido los siguientes pasos:

1. **Título del *Product Box***: inicialmente, se pensó un título para el *Product Box*, que en este caso guarda una clara relación con el título del trabajo, y a la vez representa claramente lo que es el sistema desarrollado.
2. **Metáfora gráfica**: el *Product Box* está pensado como una herramienta muy gráfica, por lo que es importante la presencia de una metáfora visual que represente las ventajas del sistema. En este caso, se ha elegido como metáfora una imagen que representa a un posible estudiante con dudas, que tras el uso del sistema de tutorías mediante las aplicaciones desarrolladas en este trabajo, consigue alcanzar sus éxitos académicos.
3. **Eslogan**: en un anuncio, es importante tener un eslogan que enganche a los posibles usuarios al sistema. En este caso, se ha seleccionado el siguiente eslogan: "Reserva tus tutorías, en cualquier lugar, en cualquier momento"

Figura 2: Product Box

4.1.4- NOT List

El objetivo de esta tarea consiste en obtener aquellas características de alto nivel que nuestro sistema tendrá una vez desarrollado. De esas características, saldrán las historias de usuario que serán cubiertas por los diferentes sprints. Asimismo, con esta herramienta también se pretende dejar claro que características **no** tendrá nuestro sistema, y cuales han quedado en un estado de **no resolución**, ya que no se ha llegado a tomar una decisión definitiva sobre si incluirlas o no en el sistema.
Como resultado de este paso del *Inception Deck* se pretende obtener un listado de características (identificadas con el formato FX, siendo X el número de la característica), repartidas en las siguientes categorías:

- **Características del proyecto (IN)**: aquello que se va a hacer, que se tiene como objetivo conseguir. Contendrá las características que se pretende implementar en el sistema.
- **Características que están fuera del proyecto (OUT)**: especificación clara de que características no va a cubrir el proyecto, bien porque se haya tomado la decisión de posponerlas a futuras versiones del proyecto, o simplemente porque quedan fuera del ámbito del sistema. En cualquiera de los casos, son características de las cuales no hay que preocuparse.
- **Características no resueltas (UNRESOLVED)**: en esta sección estarán aquellas características sobre las que aún no se ha tomado la decisión de incluirlas o no entre los objetivos del proyecto. Se deben mover todas las características de esta sección a la secciones IN o OUT en cuanto al equipo le sea posible tomar la decisión sobre ellas.

Con esta distinción, se consigue centrar la atención en los objetivos que realmente se quieren conseguir, impidiendo en gran medida que el tiempo de desarrollo se pierda en implementar funcionalidad que realmente no es objetivo del proyecto.
Una *NOT List* aporta al cliente una visión clara de lo que se va a hacer y lo que no, lo cual es útil en múltiples situaciones como, por ejemplo, a la hora de realizar el presupuesto del proyecto, o como apoyo para la estimación del tiempo de duración de los sprints. Puesto que va a ser una lista que va a trabajarse con el cliente del producto, solo contendrá características a un nivel muy alto, a ser posible con apenas detalles técnicos.

Tras sendas reuniones mantenidas con el tutor del TFG, en las que se debatieron los requisitos que cubrirían las aplicaciones desarrolladas en este trabajo, se obtuvo como conclusión la siguiente *NOT List*:

NOT LIST	
IN	**OUT**
F1: Autenticación mediante usuario y contraseña, protección de los datos mediante un sistema de seguridad. **F2**: Control de sesiones en la aplicación cliente Android. **F3**: Consultar reservas de tutorías, y tutorías ya completadas. **F4**: Actualización manual de los datos de la aplicación. **F5**: Reservar y cancelar una tutoría. **F6**: Registrar una tutoría completada.	**F7**: Notificaciones automáticas por correo. **F8**: Notificaciones automáticas en la aplicación. **F9**: Establecer el horario inicial de tutorías semanales.
UNRESOLVED	
F10: Modificar una reserva de tutoría.	

Figura 3: NOT List

4.1.5- Show your solution

Una vez llevados a cabo todos los pasos del *Inception Deck* relacionados con el contexto del proyecto, es necesario empezar a enfocar a un nivel más técnico **qué** tipo de sistema se va a desarrollar, y **cómo** va a desarrollarse.

En este paso del *Inception Deck* se pretende, tras el estudio que ya se ha realizado sobre los requisitos que va a cubrir el sistema, diseñar un diagrama que represente la arquitectura de dicho sistema al completo, indicando los componentes de los que va a constar, y las relaciones que hay entre ellos. Este diagrama no busca ser un diagrama orientado a cualquier persona sin un conocimiento técnico específico, sino un esquema con un relativo nivel de detalle técnico. Por tanto, no es un diagrama pensado para que lo lea y trabaje con él nuestro cliente, sino nuestro equipo de desarrollo.

A pesar de que, debido a la particularidad de este trabajo, no existe un equipo de desarrollo como tal, si existe un cliente (en este caso correspondería al tutor del trabajo) y un desarrollador (el alumno), ambos con el nivel de conocimiento técnico suficiente como para que esta herramienta siga manteniendo su utilidad, especialmente a la hora de planificar las fases del desarrollo o *sprints*.

El esquema representativo de la arquitectura del sistema que se ha propuesto para este trabajo es el siguiente:

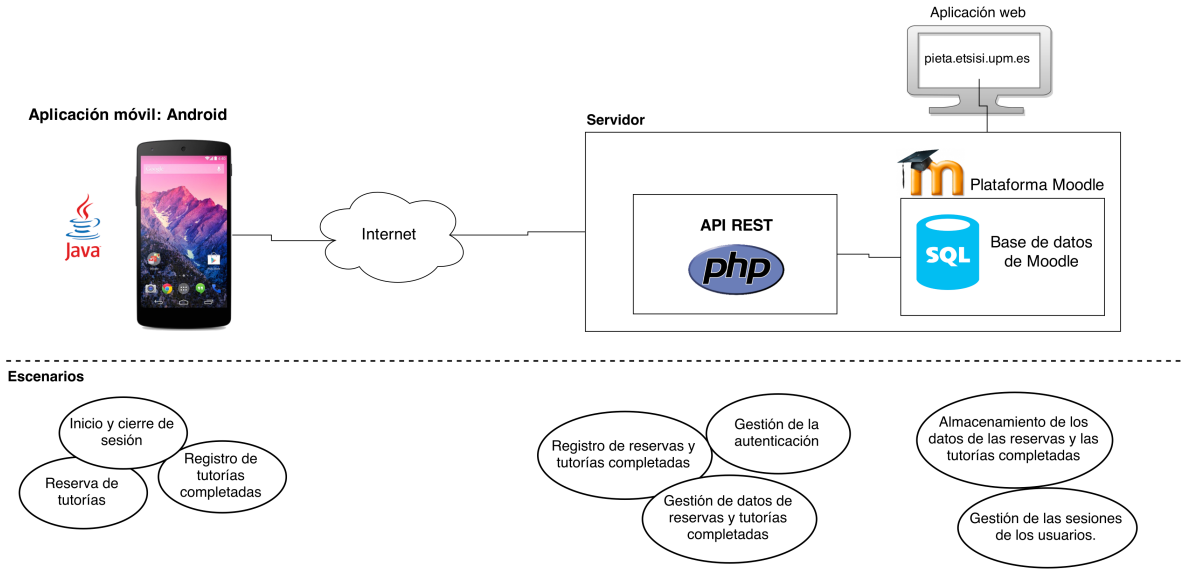

Figura 4: Esquema representativo de la arquitectura del sistema

Tal y como puede apreciarse en la figura 4, la arquitectura cliente-servidor será la que seguirá este sistema. El núcleo de la aplicación consistirá en un plugin del framework Moodle, en cuya base de datos se integrarán las tablas correspondientes a la aplicación de gestión de tutorías. Los clientes operarán contra esa base de datos, de distinta manera dependiendo del cliente:

- Aplicación web: la aplicación web actual dejará de existir, y se pasará a utilizar la interfaz web de Moodle para manejar la aplicación. Al sustituirse la aplicación web por un plugin de Moodle, que ya estará integrado con la interfaz de Moodle, el manejo será mucho mas cómodo, especialmente a la hora de gestionar usuarios y sus roles.
- Aplicaciones móvil: las aplicaciones móvil son independientes al framework Moodle, por lo que no tienen interacción directa ni con él, ni con su base de datos. Por tanto, para que las aplicaciones puedan trabajar contra esa base de datos, se ha desarrollado una API Rest, alojada en el mismo servidor que la aplicación Moodle, que encapsula la funcionalidad de las aplicaciones móvil, y les proporciona una interfaz con la que trabajar contra la base de datos, sin tener acceso explícito a ella.

La sección de escenarios recogida en el esquema menciona las características y funcionalidades que, a nivel alto, soportará el sistema, así como en que parte de la arquitectura estarán emplazadas. Por ejemplo, pueden verse los escenarios relativos al sistema de autenticación y seguridad del sistema (que será descrito mas adelante en esta memoria), repartidos entre la aplicación móvil (que proporciona las opciones de inicio y cierre de sesión), el backend (que gestiona la autenticación y seguridad de las peticiones), y la base de datos (que gestiona las sesiones de los usuarios autenticados).

4.1.6- Ask what keeps us up at night

En este paso del *Inception Deck* se tiene como objetivo estudiar todos aquellos posibles riesgos con los que el equipo puede encontrarse durante el desarrollo del proyecto. Una vez identificados, será mas fácil para el equipo estar al tanto de lo que **no** se debe hacer, de cara a evitar caer en estos riesgos.

A la hora de afrontar un proyecto, hay que tener presente que los problemas pueden llegar por cualquier frente, y en cualquier momento. Al comienzo del proyecto es cuando mas libertad de acción se tiene, por lo que es vital tratar de anticiparse a los problemas que puedan surgir, para tomar desde el principio las decisiones adecuadas de cara a evitarlos.

El proceso a seguir en la identificación de riesgos es el siguiente:

- **Brainstorm de posibles riesgos**: tanto el equipo de desarrollo como el cliente deben mantener una sesión de brainstorming en la que salgan a la luz todos los posibles riesgos a los que se enfrenta el proyecto. La presencia del cliente es vital, ya que puede aportar a la lista posibles riesgos sobre los que el equipo no tiene control, especialmente relacionados con motivos económicos.
- **Categorización de riesgos**: una vez identificados los posibles riesgos, deben separarse en dos categorías: aquellos riesgos que merece la pena abordar, y aquellos que, bien por no ser prioritarios, o por no tener el equipo capacidad para abordarlos, no van a tratarse en un primer lugar.

Para este proyecto en particular, se identificaron los siguientes riesgos principales:

- **Falta de tiempo**: debido a la relativamente poca cantidad de tiempo para la realización del trabajo, aproximadamente 4 meses, desde el principio la falta de tiempo para concluir el trabajo ha sido un riesgo de una alta prioridad. Para evitar este riesgo, se ha hecho un esfuerzo extra a la hora de repartir bien el trabajo a realizar en los distintos sprints, tratando de aprovechar al máximo el tiempo disponible del único desarrollador.
- **Volumen de trabajo excesivo**: al riesgo de la falta de tiempo se suma la carga de trabajo excesiva. En este proyecto inicialmente se pretendía desarrollar tres aplicaciones (API Rest y aplicaciones móvil Android e iOS), en un tiempo bastante limitado, y con los retrasos que conllevan los errores imprevistos que surgen durante el desarrollo. De cara a evitar este riesgo, para una primera versión del proyecto (la correspondiente a este trabajo) se pospone la aplicación para dispositivos iOS, quedando para una primera versión la API Rest, que contiene la lógica real de negocio, y la aplicación Android, aplicación cliente que podrá ser utilizada por usuarios finales.

- **Dependencias externas**: ambas aplicaciones, especialmente la aplicación Android, tienen una cantidad considerable de dependencias externas, librerías desarrolladas por terceros de las cuales dependen gran parte de la funcionalidad crítica del sistema. Así mismo, la API Rest depende de que el plugin de Moodle con cuya base de datos trabaja esté disponible, en su versión de producción, durante el desarrollo del trabajo, ya que de esa base de datos contiene toda la información del sistema (usuarios, tutorías reservadas y completadas, horarios de profesores para tutorías, etc.). Al no ser competencia de este trabajo el desarrollo del plugin, es otra dependencia externa importante.
- **Cambios de requisitos**: un riesgo presente en cualquier proyecto es la modificación por parte del cliente de los requisitos del sistema en medio de la fase de desarrollo. En este trabajo se ha abordado este riesgo acordando entre el alumno y el tutor (que también hace las veces de product owner) desde un primer momento los requisitos del sistema a desarrollar, y estableciendo como compromiso no modificarlos durante el desarrollo del trabajo, de cara a favorecer una mejor distribución tanto del tiempo disponible como de la carga de trabajo.

Así mismo, se identificaron también riesgos mas secundarios, que no podían ser abordados o no merecía la pena abordar:

- **Indisposición del desarrollador**: al solo disponer de un desarrollador, había un considerable riesgo de que el proyecto quedase parado por indisposición del mismo por enfermedad o motivo similar. No obstante, al ser un riesgo ante el cual no se puede tomar ningún tipo de medida, no se aborda y tan solo se incluye en la lista.
- **Desaparición del proyecto**: a pesar de ser un riesgo altamente improbable, el correcto desarrollo de este trabajo depende completamente de la existencia del proyecto PIETA, por lo que su cancelación es un riesgo que está presente en todo momento. No obstante, al igual que el riesgo anterior, no se puede tomar ningún tipo de medida desde las competencias de este trabajo para prevenirlo, por lo que tan solo se incluye en la lista.

4.1.7- Size it up

En este paso del Inception Deck, se realiza una de las tareas de conceptualización que mayor efecto tendrán en la estructuración del proceso de desarrollo. Como resultado de este proceso, se obtiene una **hoja de ruta** o **roadmap** a seguir por el equipo de desarrollo.
Con esta hoja de ruta, se puede llevar a cabo un proceso de estimación de la duración total del proyecto, que pueda presentarse al cliente no como una planificación detallada y firme de los tiempos de desarrollo del proyecto, sino como una estimación a alto nivel, que como tal puede sufrir variaciones.

Esta estimación se rige por un claro objetivo: **reducir** al máximo posible los **tiempos de desarrollo**. Las metodologías ágiles abogan que cualquier proyecto tiende a ser más vulnerable a riesgos cuanto mayor es su longitud en el tiempo.

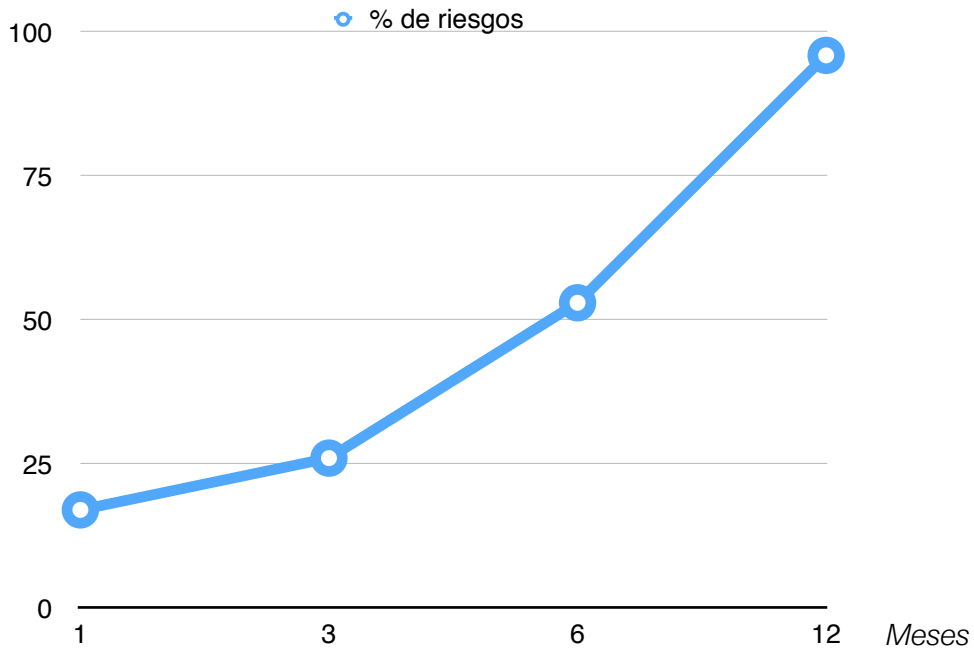

Figura 5: Porcentaje de riesgos que afectan a un proyecto en base a su duración.

En la gráfica de la figura 5, extraída del libro *"Agile Samurai"*, queda representada la teoría de las metodologías ágiles, apreciándose como el porcentaje de riesgos que afecta a un proyecto de desarrollo aumenta conforme mayor es la duración de dicho proyecto.

Las metodologías ágiles aconsejan como cota superior para la duración de un proyecto un periodo de **6 meses**. Vista la gráfica, los 6 meses de duración se corresponden con un impacto en el proyecto de, aproximadamente, el 50% de los

riesgos que puedan afectarle, un porcentaje que en ningún caso es aconsejable superar.
Este periodo es razonable como cota superior en proyectos de corta duración, en los cuales es relativamente sencillo llevar a cabo una estimación que sufra pocas variaciones, y que cumpla sin grandes problemas los plazos de entrega. Sin embargo, a la hora de realizar estimaciones de proyectos de desarrollo de sistemas complejos, es muy común que tanto los equipos de desarrollo como los demás stakeholders del proyecto, en especial los clientes, **subestiman** las características que tendrá el producto final, y en consecuencia se realiza una estimación incorrecta de la duración del proyecto. Estas **infradimensionadas** estimaciones comúnmente desembocan en **retrasos** en la entrega de los productos finales, provocados principalmente por complicaciones surgidas a lo largo del desarrollo, que retrasan las previsiones y por tanto las fechas de entrega.

Sin embargo, es obvio que no todos los proyectos de desarrollo pueden abordarse en un periodo máximo de 6 meses, por lo que habrá muchos proyectos cuya duración se estime por encima de esos 6 meses.

Como solución a este problema, las metodologías ágiles proponen **fragmentar** el producto final a desarrollar en pequeños componentes mas manejables, que no superen los 6 meses de tiempo de desarrollo (y, a ser posible, tampoco los igualen). Con esta división, se pretende enfocar el proyecto no como un todo indivisible, sino como un conjunto de componentes independientes (dentro de lo posible) cuyo tiempo de desarrollo sea considerablemente menor, y pueda estimarse de forma mas precisa.

Con esta división, se pretende conseguir ventajas como:

- **Reducción del porcentaje de riesgos**: siguiendo la gráfica anterior, desarrollando el producto en componentes independientes, se consigue una estimación mas precisa, que reduce los efectos de los riesgos sobre el proyecto.
- **Uso de la metodología de trabajo por entregas**: al dividir el producto en componentes, el cliente no recibe una única entrega final del producto que ha solicitado. En su lugar, el equipo de desarrollo entrega al cliente componentes parciales de su sistema final, lo mas independientes posibles entre si. Este sistema de trabajo conlleva ciertas ventajas atractivas tanto para el cliente como para el equipo de desarrollo:
  - En caso de que el equipo de desarrollo contratado finalice su relación contractual con el cliente, por cualquier motivo posible, el cliente tendrá en su poder los componentes que ya hayan sido desarrollados. Por tanto, podrá

contratar otro equipo de desarrollo que partirá del trabajo avanzado por el primer equipo. Esto reducirá considerablemente el retraso que de por si supone un cambio de equipo de desarrollo.

- En caso de que el cliente decida suspender el desarrollo del producto, o el proyecto en cuestión pierda los fondos que tuviese disponibles, el equipo de desarrollo también se ve beneficiado por el trabajo por entregas. Al haber fragmentado el trabajo, el equipo puede cobrar los componentes de forma independiente, lo que les asegura que, independientemente de la viabilidad del proyecto, el equipo habrá cobrado por aquellos componentes que ya hayan desarrollado.

Vistas las ventajas anteriormente expuestas, es preceptivo llevar a cabo el desarrollo de la **hoja de ruta** de nuestro sistema, que determinará su duración estimada, y dará pie a tomar las medidas de fragmentación que se consideren necesarias para su mas correcto desarrollo.

Es vital contar con la NOT List como punto de partida para el desarrollo del roadmap, ya que las características que va a tener nuestro sistema, y que por tanto habrá que desarrollar e incluir en la hoja de ruta, son aquellas que se han incluido en la sección IN de la NOT List.

Partiendo de este listado de características, el objetivo es estructurar el desarrollo en una serie de entregables o *releases* del producto final, que vayan aportando funcionalidad completamente desarrollada al cliente. Este sistema es acorde con la expuesta metodología de trabajo por entregas.
Cada una de estas releases deberá tratar una o varias de las características que tendrá nuestro producto final, implementándolas completamente. Para obtener el rendimiento mas óptimo posible de esta distribución, se deben tener en cuenta los siguientes factores:

- La **duración** debe ser similar para todas las releases: establecer una duración fija y similar para todas las releases consigue que el ritmo de trabajo del equipo sea mas predecible, y su flujo de trabajo continuo. Puede encontrarse una explicación mas detallada en artículos como el redactado por la compañía *Agility Software* [7].
- El **valor de negocio** que aporta cada release debe intentar reducirse progresivamente: a la hora de afrontar un desarrollo mediante esta metodología, se debe tratar de implementar en primer lugar aquella funcionalidad que aporte mayor valor de negocio al usuario final.

- El **esfuerzo** debe mantenerse constante con cada release: de cara a mantener lo mas similares posibles las releases, se debe tratar de que el esfuerzo total destinado a cada release sea constante.

Valor de negocio

La metodología usada para establecer el valor de negocio de cada característica consiste en repartir un valor fijo de puntos, en este caso 100, entre todas las características, considerando que tienen mas valor de negocio con respecto a las demás aquellas que tengan mayor puntuación.

Punto-característica y esfuerzo

El valor del esfuerzo para cada característica se asigna utilizando el convenio **punto-característica**, que consiste en una relación de puntos para cada característica, en la cual se considera que es necesario un mayor esfuerzo para aquellas características mas puntuadas.

Historias de usuario

Como paso previo a la descripción del proceso de desarrollo mediante sprints, es necesario definir las historias de usuario que especificarán los casos de uso relacionados con las características a cubrir por el sistema que va a desarrollarse.

Puesto que ya se ha llevado a cabo parte del exhaustivo proceso de estudio del proyecto que supone el *Inception Deck*, pueden utilizarse los resultados del mismo para generar las historias de usuario. Particularmente la *NOT List* es un excelente punto de apoyo, ya que proporciona el listado de características que deben ser tratadas entre todas las historias de usuario de la aplicación.

A cada historia se le asigna una puntuación simbólica, expresada mediante puntos-historia o *story points (SP).* Esta puntuación tiene como utilidad poder conocer la velocidad real del equipo de desarrollo en cada sprint, a través de una gráfica denominada *burndown chart*, en la cual se representa el avance en el desarrollo de los SP con respecto al tiempo.

A la vista de las características obtenidas en la NOT List, y tras estudiar las posibles historias para ambos perfiles de usuario, se obtuvo como listado final el siguiente:

Historias de usuario: como usuario, espero…	Story points (SP)
US1: Iniciar sesión en múltiples dispositivos, de forma simultánea.	3
US2: Contar con un sistema de seguridad que proteja los datos de la aplicación.	7
US3: Iniciar sesión en la aplicación, mediante mis credenciales de acceso.	4
US4: Cerrar mi sesión en la aplicación.	3
US5: Cierre de sesión automático pasado un tiempo.	4
US6: Consultar un listado de mis reservas de tutorías.	5
US7: Consultar un listado de mis tutorías completadas.	5
US8: Ordenar los listados por fecha u orden de reserva.	4
US9: Actualización de los listados de reservas y tutorías completadas.	5
US10: Reservar una tutoría.	8
US11: Cancelar una reserva de tutoría.	3
US12: Consultar horas disponibles para reservar una tutoría.	4
US13: Registrar una tutoría completada.	5

Figura 6: Listado de historias de usuario

Dentro de este listado de historias de usuario, cabe destacar la presencia de una **historia de usuario épica**, que se han subdividido en varias historias en la relación final. Esta historia épica se corresponde con las características F1 y F2, relacionadas con la seguridad del sistema. Implementar el sistema completo de seguridad requiere de una cantidad de trabajo excesiva para ser considerada en una única historia de usuario, por lo que se ha subdividido finalmente en 5 historias: US1, US2, US3, US4 y US5.

Historias de documentación

Todo sistema en producción debe tener una buena documentación respaldándolo, que permita mantenerlo y/o mejorarlo cuando se considere oportuno. Con cada avance que se realice en la implementación del sistema, se debe añadir y/o actualizar la documentación correspondiente.

En esta sección se han definido aquellas historias relacionadas con las tareas de documentación a llevar a cabo. Dichas tareas serán cubiertas en los distintos sprints del proyecto, en paralelo con las historias de usuario.

Historias de documentación
DS1: Documentación individual de cada sprint.
DS2: Documentación de la API.
DS3: Documentación de la aplicación Android.

Figura 7: Listado de historias de documentación.

Este listado de historias de documentación presenta algunas particularidades:

- La historia de documentación “Documentación individual de cada sprint” se repetirá en cada sprint, ya que cada sprint se documenta individualmente en esta memoria.
- Las historias de documentación referentes a las aplicaciones backend (API) y Android se repetirán en varios sprints, ya que el desarrollo de ambas aplicaciones no se puede cubrir en un único sprint.

Hoja de ruta

Una vez se tienen claramente definidas las características que tendrá el sistema, se debe decidir con el cliente en cuantas entregas se va a dividir el proceso de desarrollo, así como las características a cubrir en cada entrega. Para llevar a cabo una mejor distribución, se deben asignar los valores de negocio y de esfuerzo a todas las *features* a implementar, siguiendo las instrucciones detalladas en el apartado anterior.

Tras estudiar la distribución de las características en releases con el cliente, queda finalmente como sigue:

Release	Característica	Valor de negocio	Esfuerzo
R1	F1	20	6
	F2	5	4
R2	F3	20	6
	F4	15	4
R3	F5	30	7
	F6	10	3

Figura 8: Distribución de releases y features del sistema, con sus valores de negocio y esfuerzo.

Una vez realizada esta distribución, se puede pasar a terminar el roadmap planificando los distintos sprints en los que se dividirá la fase de desarrollo del sistema. En estos sprints, de 2 semanas de duración, se dará cobertura a todas las características a implementar.

Si bien en el apartado anterior se puede ver un listado aplanado de todas las historias de usuario, estas historias se han obtenido a partir de las distintas características de nuestro sistema, identificadas en la NOT List. La relación entre las historias y las características del sistema es la siguiente:

Característica	Historias de usuario
F1	US1, US2
F2	US3, US4, US5
F3	US6, US7, US8
F4	US9
F5	US10, US11, US12
F6	US13

Figura 9: Relación entre las features del sistema y las historias de usuario.

Finalmente, conociendo esta relación entre características e historias de usuario, se puede llevar a cabo la división de las releases en sprints, a los que se les asignarán historias de usuario que deberán cubrir.

La distribución queda como sigue:

Release	Sprints	Historias de usuario
R1	S1	US1, US3, US4
	S2	US2, US5
R2	S3	US6, US8
	S4	US7, US9
R3	S5	US10, US12
	S6	US11, US13

Figura 10: Listado de sprints.

Con esta última distribución, quedaría completamente definida la hoja de ruta. Cabe destacar que la duración de cada release será de 1 mes, por lo que se consigue el objetivo de que la duración sea similar entre todas las releases. Asimismo, se comprueba la viabilidad del proyecto, de una duración estimada de 3 meses, con respecto al tiempo disponible de desarrollo, aproximadamente 4 meses.

Cabe destacar que, si bien la estimación encaja en el tiempo disponible, la diferencia entre la duración estimada y el periodo máximo no es excesiva, lo cual deja un relativamente escaso margen de reacción en caso de surgir problemas imprevistos durante el desarrollo del proyecto, y aumenta el riesgo siempre presente de incumplimiento de plazos de entrega.

Así mismo, en esta estimación se comprobó que la decisión tomada en el paso “Ask what keeps us up at night” del *Inception Deck*, por la cual se pospuso el desarrollo de la aplicación móvil para dispositivos iOS, fue una decisión acertada, ya que de añadirse a la estimación (con un tiempo de desarrollo similar al estimado para la aplicación para dispositivos Android), el tiempo total de desarrollo superaría los 4 meses, y por tanto no se podría garantizar la viabilidad del proyecto, menos aun considerando que estas estimaciones suelen quedarse cortas con respecto a la duración que finalmente tiene el proyecto.

4.1.8- Be clear on what's going to give

El objetivo de esta herramienta es establecer una jerarquía de prioridad entre una serie de características asociadas al proyecto.
La existencia de esta herramienta es debida a la clara presencia de conflictos frecuentes entre algunas de las características clave del proyecto. Mediante el uso de ese proceso, se trata de obtener un balanceo entre los distintos niveles de prioridad de dichas características, lo que permite establecer de una forma bastante clara en que aspectos va a centrarse especialmente el proyecto.

Dentro de las características a considerar en este balanceo, Jonathan Rasmusson destaca 4 que están por encima del resto, ya que son transversales a cualquier tipo de proyecto. A continuación se listan y describen:

- **Tiempo**: el tiempo del que se dispone para el desarrollo de cada proyecto siempre es limitado, es tarea del equipo de desarrollo utilizarlo de la mejor manera posible, tratando de evitar a toda costa retrasos en las fechas de entrega, que en consecuencia generen descontento en el cliente.
- **Presupuesto**: al igual que el tiempo, es limitado, y en general suele ser escaso. A la hora de contratar a un equipo de desarrollo, el cliente siempre tratará de minimizar el gasto que le suponga el desarrollo, limitando el presupuesto todo lo posible. El equipo debe gestionarlo de forma adecuada, y saber priorizarlo frente a otros aspectos según el tipo de proyecto que se esté abordando.
- **Calidad**: parafraseando a Rasmusson, cualquier ganancia en velocidad, resultante de una reducción en la calidad del producto final, no es mas que una falsa ilusión. A largo plazo, reducir el nivel de calidad del producto puede acarrear graves consecuencias (descontento de los usuarios, mayores costes en mantenimiento, etc.), por lo que no es una opción que el equipo de desarrollo se deba plantear. Al igual que el tiempo y el presupuesto, la calidad debe ser un factor fijo, y que siempre debe orientarse al máximo posible.
- **Alcance**: el alcance es la única de estas 4 características del proyecto que puede (y debe, en caso de necesidad) variar. La planificación del proyecto es algo volátil, que cuando la situación así lo requiera puede modificarse. Factores como la asignación presupuestaria o la fecha límite de entrega fijada por el cliente son fijos, pero en caso de que no haya tiempo o fondos suficientes para completar el proyecto, siempre puede reestructurarse, reducirse la funcionalidad a implementar, o tomar alguna solución similar que implique la modificación del alcance del proyecto.

A la vista de estos 4 pilares sobre los que se apoya la gestión de cada proyecto, se obtiene como conclusión que el **tiempo**, el **presupuesto**, y la **calidad** del producto final deben ser **fijos**, mientras que el **alcance** del proyecto es **variable**, y puede ser modificado a la hora de solucionar imprevistos.

Tras haber descrito las características, se pasa a evaluar, para el proyecto en cuestión, el orden de prioridad entre ellas, así como el nivel de flexibilidad que se va a tener para cada una. Para hacer esta tarea más cómoda y gráfica, puede hacerse uso de deslizadores, en los que uno de los extremos represente la máxima flexibilidad con respecto a esa característica, y el otro una mínima o ninguna flexibilidad.

La estructura del deslizador es la siguiente:

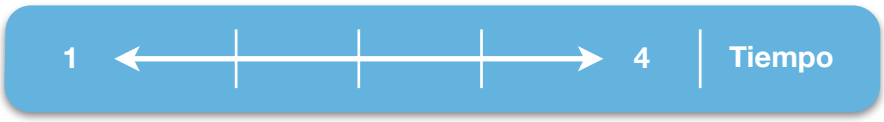

Figura 11: Diseño del deslizador de valoración de características.

Para cada característica, se debe situar el deslizador en una de las secciones, teniendo en cuenta las siguientes reglas:

- Las características no pueden tener todas el mismo nivel de importancia.
- En general, el mismo nivel **no** puede estar ocupado por 2 o más características.
- Se debe definir de forma clara el significado de los extremos del deslizador. Para este caso particular, el valor 1 implicará un alto nivel de importancia, y por tanto un alto grado de inflexibilidad con respecto a esa característica, mientras que el valor 4 indicará un menor nivel de importancia, y por tanto una mayor flexibilidad.

Es importante destacar que el hecho de que una característica tenga un valor menor que otra no implica que no sea importante, sino que hay otras prioridades por encima de ella en el proyecto.

En un proyecto tipo de desarrollo, a la hora de utilizar esta herramienta se pediría a los propios clientes que eligiesen ellos mismos el nivel de prioridad de cada característica. Sin embargo, en este caso particular, no solamente no existe un equipo de desarrollo ni unos clientes como tal, sino que es necesario tener en cuenta las siguientes premisas:

- El tiempo está 100% limitado: puesto que la Universidad delimita en su normativa las fechas límite de presentación de los trabajos de fin de grado, dichas fechas suponen el plazo máximo con el que cuenta el proyecto para ser llevado a cabo.

Por tanto, el deslizador del tiempo se situará muy cercano al 1, ya que el nivel de inflexibilidad con respecto al tiempo en este trabajo es máximo.

- No existe una asignación presupuestaria: puesto que el proyecto no deja de ser un trabajo que se desarrolla en el ámbito de la enseñanza universitaria, no requiere de asignación presupuestaria, ya que los desarrolladores (en este caso, el alumno) no perciben sueldo alguno, y los recursos utilizados son aquellos disponibles en la propia universidad. Puesto que el proyecto no está dotado de presupuesto alguno, el nivel de relevancia del presupuesto con respecto a las demás características es nulo, por lo que el deslizador se sitúa en una posición muy cercana al valor 4.

Tras aplicar esta herramienta al proyecto que ocupa este trabajo, se obtuvieron los siguientes valores en los deslizadores para cada característica:

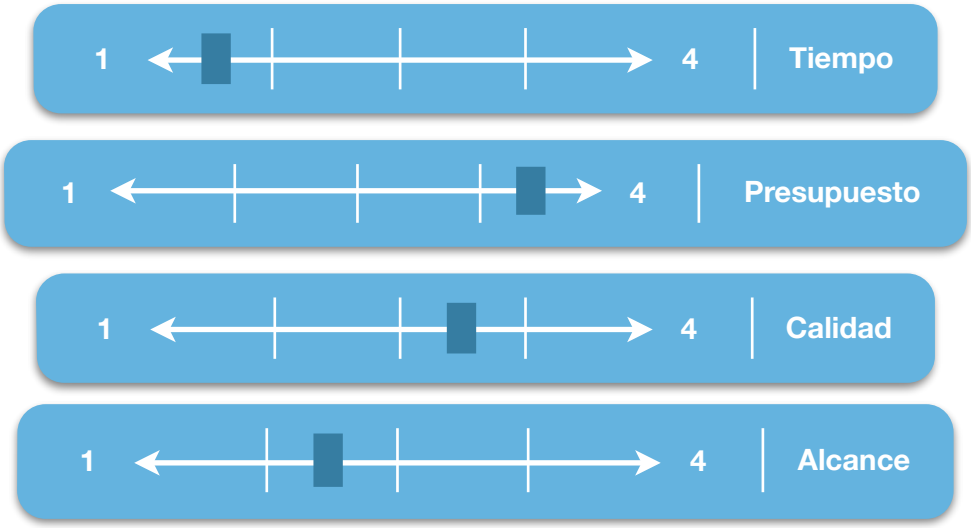

Figura 12: Valoración de las 4 principales características del proyecto.

De esta distribución, se pueden extraer las siguientes conclusiones:

- El alcance debe ser todo lo invariable posible y, en caso de variar, debe suponer un aumento en el volumen, nunca una disminución. Puesto que la particularidad de este trabajo exige al alumno un mínimo de volumen, es más que conveniente fijar en un principio el alcance total del proyecto, (estimando su viabilidad contrastando los plazos) para después no tener que modificarlo a lo largo del proceso de desarrollo.
- La calidad del producto final puede salir perjudicada con esta distribución, ya que se sitúa su deslizador en un valor que permite una relativa flexibilidad a la hora de realizar variaciones.

4.2- Sprints

En las siguientes secciones se describen los sprints acontecidos a lo largo del proceso de desarrollo del sistema, así como los resultados obtenidos.

4.2.1- Sprint 1

4.2.1.1- Sprint backlog

Al ser el primer sprint del proceso de desarrollo, una gran parte de su carga de trabajo se corresponderá con operaciones de configuración base de las aplicaciones a desarrollar. Al margen de dichas operaciones, el sprint cubrirá las historias de usuario US1, US3 y US4, todas ellas relacionadas con las tareas de autenticación en el sistema.

Product backlog		
R1	**S1**	**US1, US3, US4**
	S2	US2, US5
R2	F3	
	F4	
R3	F5	
	F6	

Figura 13: Product backlog con las historias de usuario del sprint 1 destacadas.

US1: Iniciar sesión en múltiples dispositivos, de forma simultánea

Esta historia busca dar la posibilidad a los usuarios finales de utilizar la aplicación, con sus usuarios correspondientes, en más de un dispositivo, lo que les otorga mayor libertad en su uso. Se le ha asignado a esta historia un valor de 3 SP.

US3: Iniciar sesión en la aplicación, mediante mis credenciales de acceso

Esta historia representa una de las funcionalidades básicas de la aplicación cliente, el inicio de sesión, que constará de una vista exclusiva. Se le ha asignado a esta historia un valor de 4 SP.

US4: Cerrar mi sesión en la aplicación

Historia de usuario complementaria a la anterior, cubre la funcionalidad de la aplicación cliente relacionada con el cierre de la sesión actual de un usuario. Se le ha asignado a esta historia un valor de 3 SP.

Como resultado de este sprint, se espera obtener una versión inicia del sistema, con tan solo la funcionalidad de inicio y cierre de sesión implementada.

4.2.1.2- Detalle del sprint

En este primer sprint se da por comenzada la etapa de implementación del sistema, comenzando tanto con la implementación del backend como de la aplicación cliente Android.

API

Para el desarrollo de la API, fue necesario decidir primero que tecnologías eran las mas adecuadas para este proyecto, así como que herramientas de desarrollo (IDEs) se iban a utilizar a lo largo de todo el proceso para llevar a cabo la implementación. Tras estudiar una serie de tecnologías posibles, finalmente se tomó la decisión de implementar la API usando las siguientes tecnologías y herramientas:

- **PHP** [8]: el lenguaje de programación finalmente escogido fue PHP. Se escogió este lenguaje tanto por las facilidades que otorga a la hora de desarrollar código a nivel servidor (por ejemplo, conectividad con bases de datos) como por el uso del framework Slim que se describe en el siguiente punto.
- **Slim framework** [9]: constituye el núcleo de funcionamiento de todo el backend. Este framework, orientado al desarrollo de APIs, entre otras funcionalidades, aporta al proyecto un sistema de mapeo automático de peticiones Rest a funciones asociadas:

```php
<?php
$app = new \Slim\Slim();
$app->get('/books/:id', function ($id) {
    //Show book identified by $id
});
```

Figura 14: Extracto de la documentación de Slim.

En la figura 14 se puede observar un ejemplo de este sistema de mapeo automático, por el cual se captura la petición GET /books/:id en una función que actúa de callback, procesa la petición llevando a cabo las operaciones necesarias en el lado backend del sistema, y devuelve el resultado al cliente que ha realizado la petición.

En relación con este sistema de mapeo, Slim contiene otra herramienta de gran utilidad para la aplicación, y uno de los motivos por los que se seleccionó finalmente este framework para construir la API: funciones **middleware**. Slim da la posibilidad de ejecutar funciones intermedias antes de procesarse una petición, lo cual ha sido especialmente útil a la hora de implementar el sistema de autenticación en el backend.

Todas las peticiones que la API soporta estarán controladas por una función que actúa de middleware, comprobando si la aplicación cliente que está realizando la petición está debidamente autenticada.

- **PhpStorm IDE** [10]: entorno avanzado de desarrollo de aplicaciones PHP, desarrollado por JetBrains [11], que entre otras ventajas permite configurar un sistema de autodespliegue de la aplicación en el servidor de producción, con actualización de cambios en los ficheros en tiempo real. En definitiva, un entorno muy completo, motivo por el cual se elige para este desarrollo.

Tras la selección de las herramientas y tecnologías a utilizar para el desarrollo del backend, se procedió a su instalación e implementación de las primeras peticiones de prueba.

La estructura inicial de la aplicación constaba de los siguientes componentes:

- Módulo de captura de peticiones HTTP: implementado en el fichero **index.php**, este componente contiene todos los mapeos de peticiones HTTP a funciones PHP. A lo largo de este sprint, se implementaron peticiones de prueba, de cara a comprobar el correcto funcionamiento del framework, así como una primera versión de las peticiones de inicio y cierre de sesión.
- Fichero de configuración **config.php**: contiene toda la información necesaria para la conexión con la base de datos: host en el que está alojada la base de datos, usuario de la base de datos y contraseña, y nombre de la base de datos a la que se quiere acceder.
- Fichero **htaccess**: fichero de configuración interna de Apache, configurado de manera que redireccione todas las rutas de las peticiones al fichero index.php, que encargará de capturarlas y mapearlas.
- Archivos del framework de Slim.

Base de datos

Una vez configurada la primera versión de la API, se pasó a instalar la base de datos del sistema. Puesto que el objetivo de este proyecto es que funcione en conjunción con el plugin del framework Moodle que se está desarrollando a la par que este sistema, la base de datos a utilizar será la de Moodle. A tal efecto, se configura un servidor de pruebas con una instalación de MySQL y una base de datos clonada de una instalación de Moodle, añadiendo aquellas tablas que utilizarán tanto el plugin de Moodle como la API.

A continuación puede verse una definición de las tablas a utilizar por la API (se han omitido aquellas tablas internas del esquema de datos Moodle que no van a ser utilizadas por la aplicación):

Base de datos: versión inicial	
Tablas de Moodle	**Tablas propias de la aplicación**
• mdl_user: tabla con toda la información de los usuarios registrados en el sistema Moodle. • mdl_role_assignments: tabla que contiene la información sobre la asignación de roles a cada uno de los usuarios del sistema. • mdl_course: tabla con la información de los cursos (asignaturas) del sistema Moodle. • mdl_enrol: tabla con la relación entre el ID de un curso y los distintos tipos posibles de enrolamiento que ese curso tiene (con su correspondiente ID). • mdl_user_enrolments: tabla que contiene la relación entre los IDs de los usuarios y los de sus enrolamientos en los distintos cursos registrados en la plataforma.	• mdl_tutor_schedule: tabla mediante la cual se generan las franjas horarias para tutorías de cada profesor. • mdl_tutor_timetable: tabla con la información sobre las franjas horarias de cada profesor, generadas a partir de la tabla anterior. • mdl_tutor_reserve: tabla que contiene la información de todas las reservas registradas en la aplicación. • mdl_tutor_tutorship: tabla que contiene la información de todas las tutorías completas registradas en la aplicación.

Figura 15: Esquema inicial de tablas de la base de datos.

Haciendo uso de estas tablas, la API implementa toda la lógica de negocio del sistema, desde operaciones básicas como la autenticación de las peticiones o el control de roles de lo usuarios, hasta las operaciones clave como la reserva de tutorías o el registro de tutorías completadas.

Aplicación cliente Android

En este sprint se establece la base de la aplicación, así como su estructura, sin realizar inicialmente ningún tipo de avance en cuanto a funcionalidad. Una vez configurada la base de la aplicación, se pasa a desarrollar la parte cliente correspondiente al inicio y cierre de sesión, ambas historias a cubrir por este sprint.

Las vistas desarrolladas en la aplicación Android a lo largo de este sprint son las siguientes:

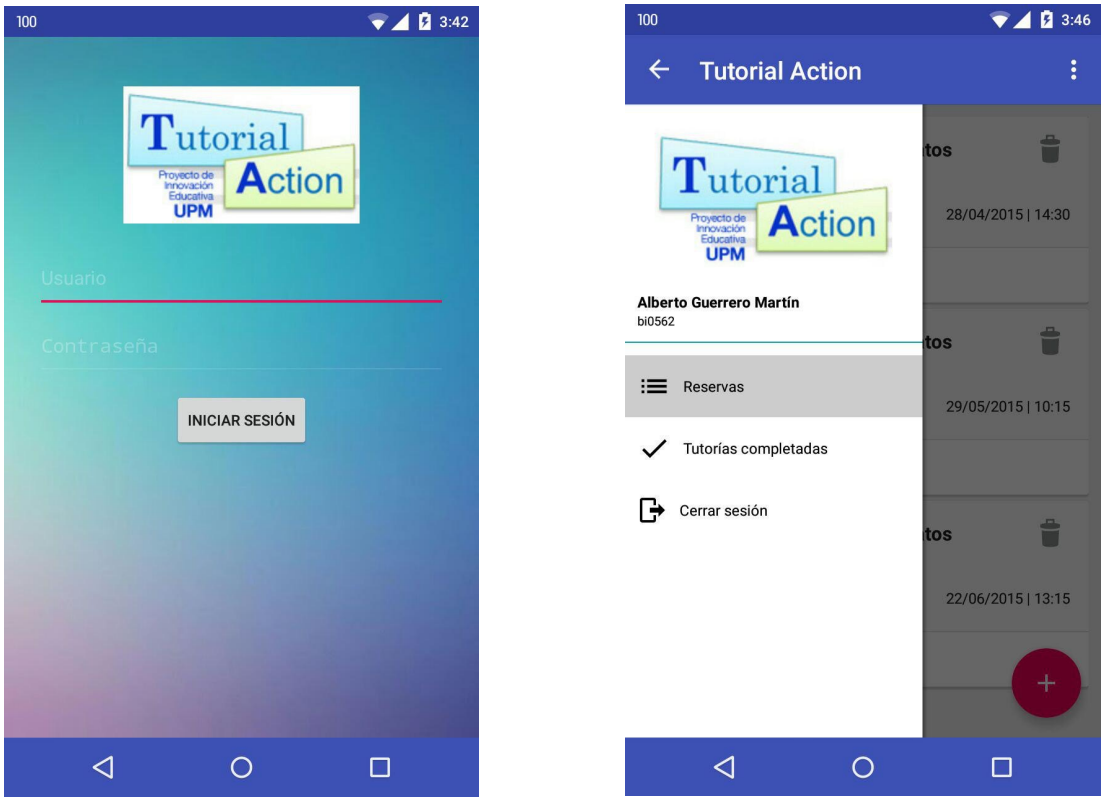

Figura 16: Volcados de la interfaz de usuario.

En la figura 16 se pueden ver la vista de inicio de sesión (izquierda) y la opción de cierre de sesión en el menú de navegación (derecha). Estas vistas hacen uso de las peticiones desarrolladas en el backend para el inicio y cierre de sesión.

Cabe destacar que el menú de navegación no es funcional tras este sprint, ya que tan solo se ha implementado la opción de “Cerrar sesión”. Las demás secciones se desarrollarán en sprints posteriores.

4.2.1.3- Modelo de datos

Al finalizar este sprint, el modelo de datos resultante es el siguiente:

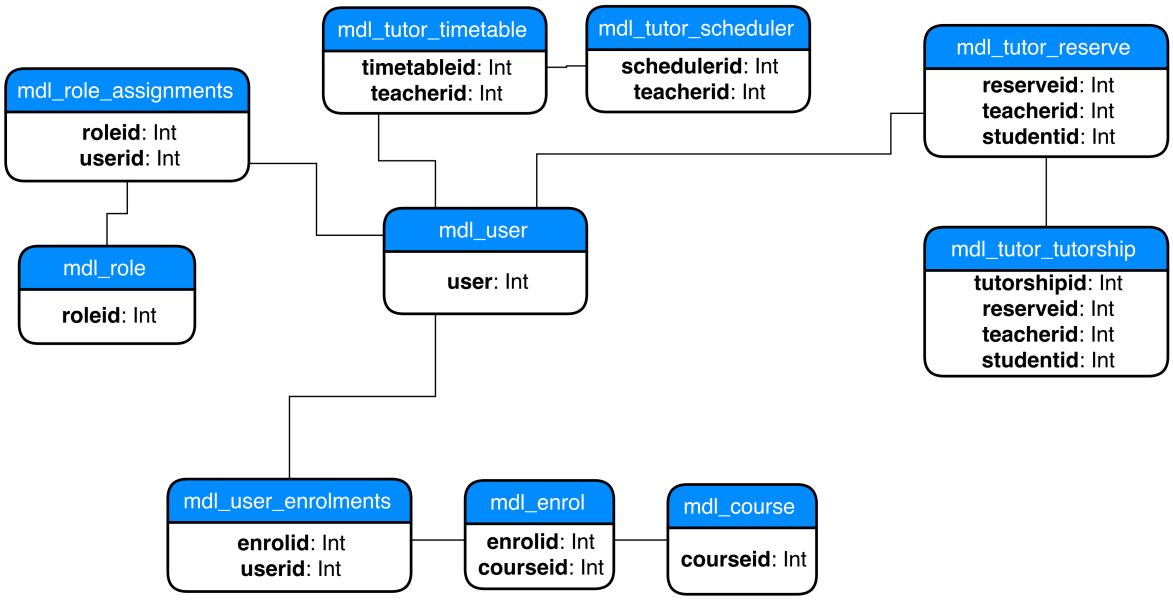

Figura 17: Modelo de datos tras el sprint 1.

4.2.1.4- Diagrama de clases

El diagrama de relación entre las clases de la API, tras este sprint, queda como sigue:

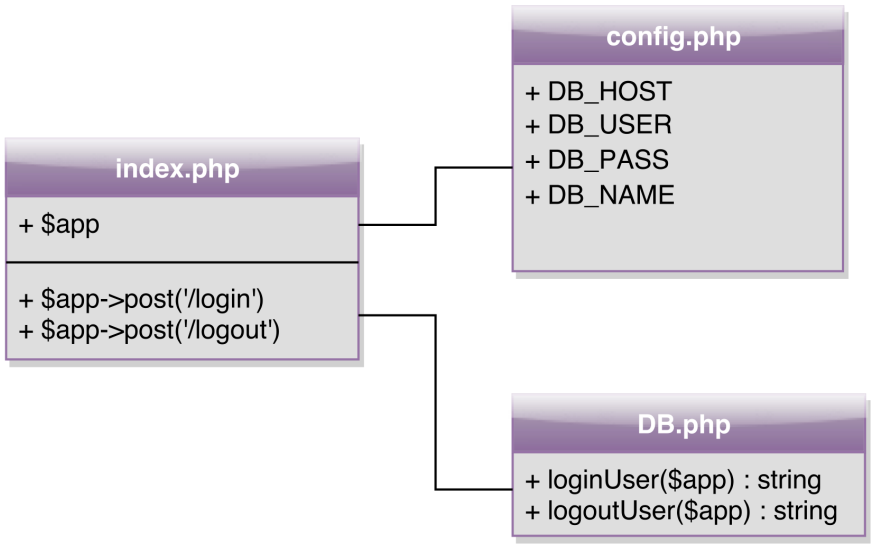

Figura 18: Diagrama de clases de la API tras el sprint 1.

En el caso de la aplicación Android, tras este sprint se obtiene una aplicación con una vista de inicio de sesión funcional, un módulo de red con las peticiones de inicio y cierre de sesión, y una vista vacía, que corresponderá a la vista principal de la aplicación tras el inicio de sesión. El diagrama queda como sigue:

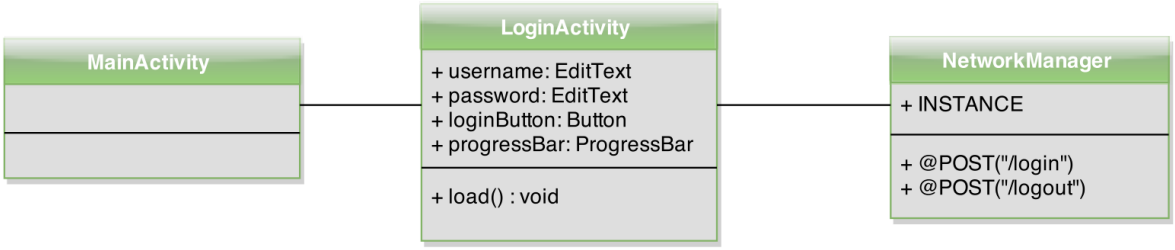

Figura 19: Diagrama de clases de la aplicación Android tras el sprint 1.

4.2.1.5- Sprint review

El resultado de este sprint es una primera versión de la API, a la que se le han añadido las peticiones definitivas de inicio y cierre de sesión; una versión bastante completa de la base de datos; y una base de la aplicación Android, que ahora consta de una vista de inicio de sesión completamente funcional, y de una opción de cierre de sesión en el menú lateral de navegación.

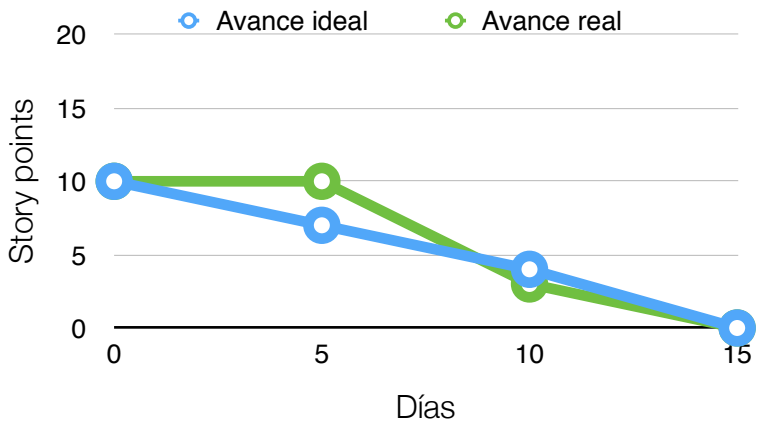

Figura 20: Burndown chart del sprint 1.

4.2.1.6- Retrospectiva

En la siguiente tabla se resumen, a modo de retrospectiva, las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El avance real en el desarrollo ha ido bastante ajustado al teórico.	Mayor número de pruebas.	Escaso control de errores a nivel de cliente.

Figura 21: Retrospectiva del sprint 1.

4.2.2- Sprint 2

4.2.2.1- Sprint backlog

En este segundo sprint, que cierra la primera release del proceso de desarrollo, se prosigue con las tareas realizadas en el anterior sprint, relacionadas con la implementación del sistema de autenticación y seguridad.

Product backlog		
R1	S1	US1, US3, US4
	S2	**US2, US5**
R2	F3	
	F4	
R3	F5	
	F6	

Figura 22: Product backlog con las historias de usuario del sprint 2 destacadas.

US2: Contar con un sistema de seguridad que proteja los datos de la aplicación

Esta historia representa todo el sistema de seguridad que debe implementarse en el sistema para garantizar que todos los datos que maneja estén debidamente protegidos, garantizándose su confidencialidad e integridad. Debido a su importancia, se le ha asignado a esta historia un valor de 7 SP.

US5: Cierre de sesión automático pasado un tiempo

Con esta historia se cubre uno de los requisitos de seguridad de la aplicación, consistente en evitar dejar sesiones iniciadas permanentemente. Se le ha asignado a esta historia un valor de 4 SP.

Como resultado de este sprint, se espera obtener una versión del proyecto que implemente completamente el sistema de autenticación y seguridad de los datos de la aplicación que se propone en este sprint, tanto a nivel de la API como a nivel de la aplicación cliente Android.

4.2.2.2- Detalle del sprint

Desde el inicio de la implementación de la API, el tratamiento de la seguridad de los datos que maneja estuvo muy presente. La base de datos almacena información confidencial sobre los usuarios del sistema, por lo que protegerla es imprescindible.
Por tanto, tras desarrollar la base del sistema de autenticación, se pasó a desarrollar el sistema de seguridad de la misma.

Seguridad de la API

Toda la API está envuelta dentro de un sistema de seguridad, pensado para proteger la privacidad de los datos tanto a nivel backend como a nivel de las aplicaciones frontend. A continuación puede verse un diagrama explicativo del sistema de seguridad implementado:

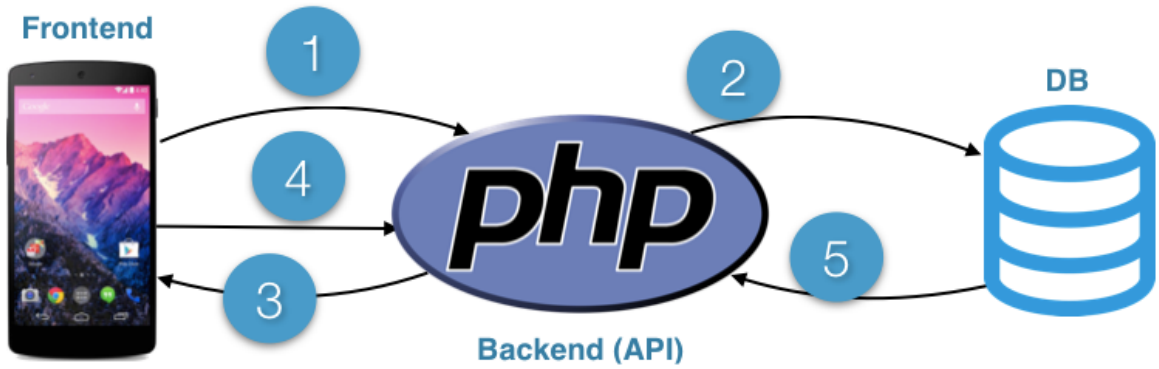

Figura 23: Diagrama representativo del sistema de seguridad.

1. **Inicio de sesión**: la aplicación cliente, en este una aplicación móvil, realiza un inicio de sesión contra el backend, aportando como datos un usuario y una contraseña. Para garantizar la transmisión segura de estos datos, es conveniente que el servidor que aloja la API tenga instalado un certificado SSL [12].
2. **Generación del token de sesión**: la API, tras recibir la petición de autenticación de la aplicación cliente, contrasta el usuario y la contraseña contra la base de datos. En caso de ser ambos datos correctos, genera un token de sesión para ese usuario, que registra en la BD en la tabla **mdl_tutor_user_token**, creada a tal efecto, que relaciona los usuarios con sus tokens de sesión.

3. **Respuesta al frontend**: tras generar el token, la API lo incluye en la respuesta a la aplicación cliente, que almacena dicho token para autenticar las futuras peticiones que realice.
4. **Autenticación de la petición**: una vez ha iniciado sesión, la aplicación puede realizar una petición a la API. Para ello, deberá incluir como cabecera de la petición el token generado en el proceso de inicio de sesión.
5. **Comprobación del token**: independientemente de la petición realizada por la aplicación cliente, la API comprueba si el token enviado junto con la petición es un token válido, es decir, si está registrado en la base de datos. En caso afirmativo, se ejecuta la petición y se devuelve la respuesta correspondiente. En caso negativo, se rechaza la petición, devolviéndose como respuesta un error de autenticación.

Para dar soporte desde el backend a este sistema de seguridad, en este sprint se realizan también los siguientes avances en el desarrollo del sistema:

- Base de datos, se incluye la tabla mdl_tutor_user_token, para almacenar los tokens generados al autenticar un usuario. Las entradas de esta tablas se borran automáticamente cada hora mediante un evento SQL, para evitar colapsar la tabla y mantener las sesiones permanentemente, lo que da cobertura a la historia de usuario US5.
- API: se modifican las peticiones de inicio y cierre de sesión, que ahora tendrán la tarea adicional de generar y eliminar los tokens de autenticación de usuarios. También se añade, para todas las peticiones, una función middleware que comprobará la validez del token incluido como cabecera de cada petición.
- Aplicación cliente Android: se desarrolla todo el sistema interno responsable de almacenar el token de sesión, de enviarlo como cabecera en cada petición que se realice a la API, y de eliminarlo tras el cierre de sesión.

4.2.2.3- Modelo de datos

Al finalizar este sprint, el modelo de datos se mantiene prácticamente similar a la versión del primer sprint. La única novedad que presenta es la inclusión de la tabla mdl_tutor_user_token, tabla responsable de almacenar los tokens de sesión generados en cada autenticación mediante usuario y contraseña que se realiza desde la API. Es en esta tabla donde la API comprobará la validez de los tokens que lleguen como cabecera de cualquiera de las llamadas.

El modelo queda como sigue:

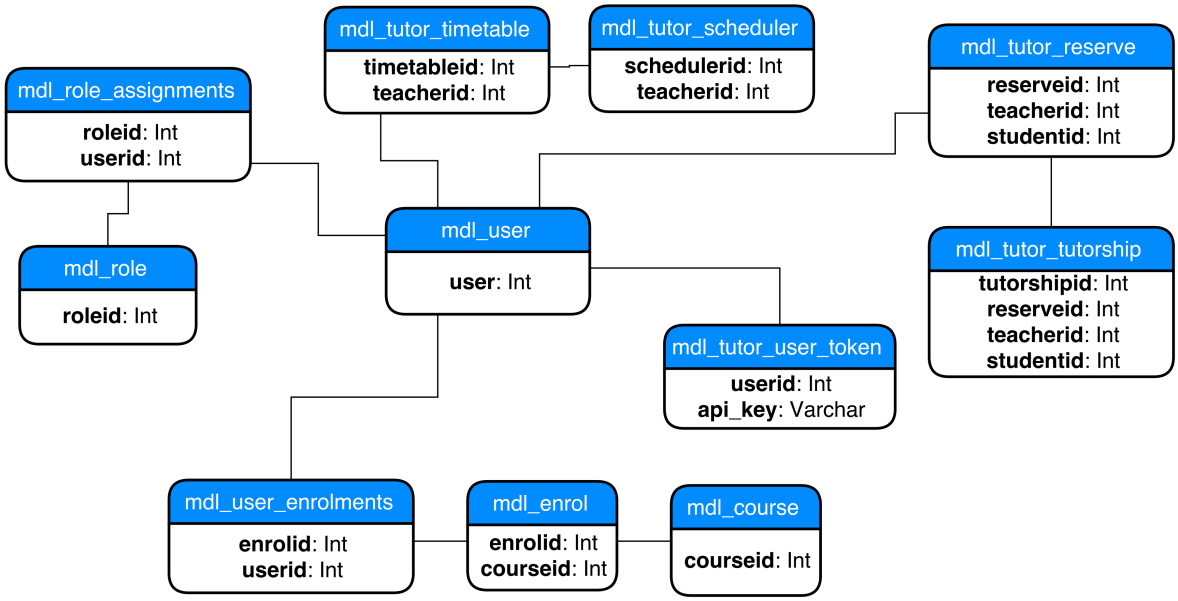

Figura 24: Modelo de datos tras el sprint 2.

4.2.2.4- Diagrama de clases

El diagrama de relación entre las clases de la API, tras este sprint, queda como sigue:

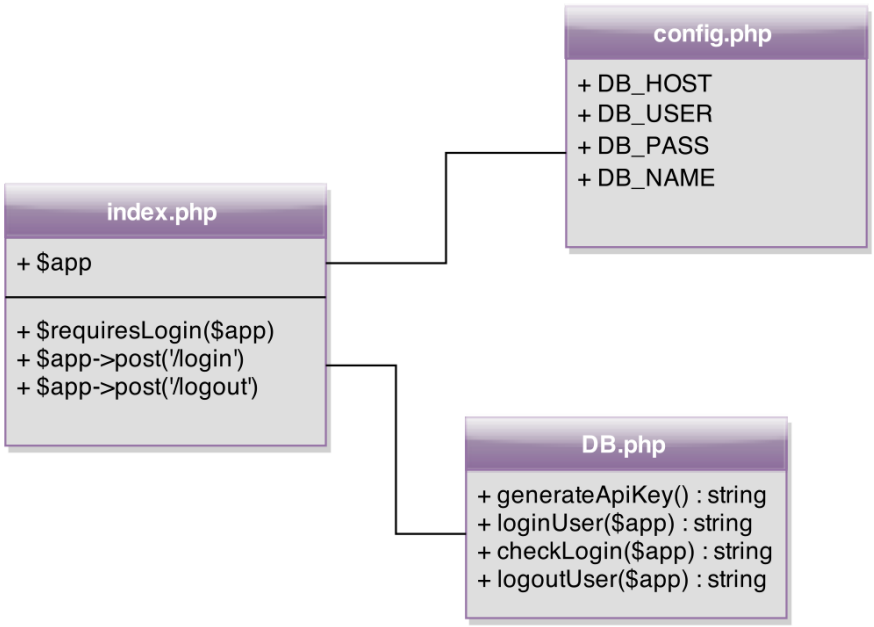

Figura 25: Diagrama de clases de la API tras el sprint 2.

En el caso de la aplicación Android, tras este sprint se obtiene una aplicación similar a la obtenida como resultado del último sprint, pero con un módulo de red que ahora controla el almacenamiento, uso y eliminación de los tokens de sesión. El diagrama queda como sigue:

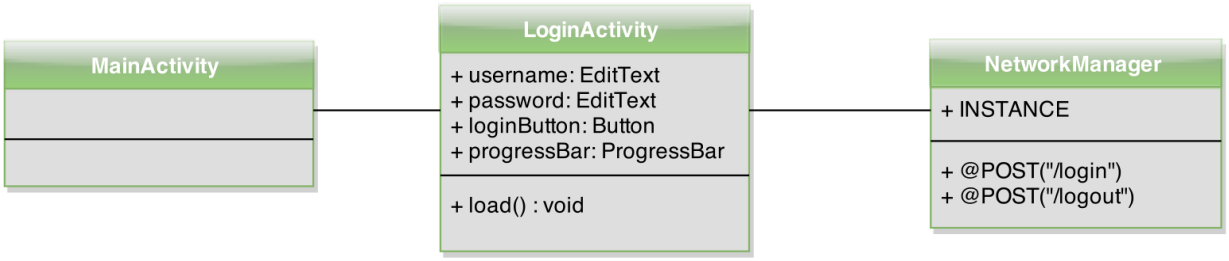

Figura 26: Diagrama de clases de la aplicación Android tras el sprint 2.

4.2.2.5- Sprint review

El resultado de este sprint es una versión de la API con un sistema de seguridad 100% implementado (a falta del uso del protocolo SSL, que no corresponde a una tare de implementación de la API); un modelo de datos en su versión final, ya que no sufrirá mas modificaciones; y una aplicación Android preparada para gestionar las peticiones a la API de forma segura, haciendo uso de los tokens de sesión.

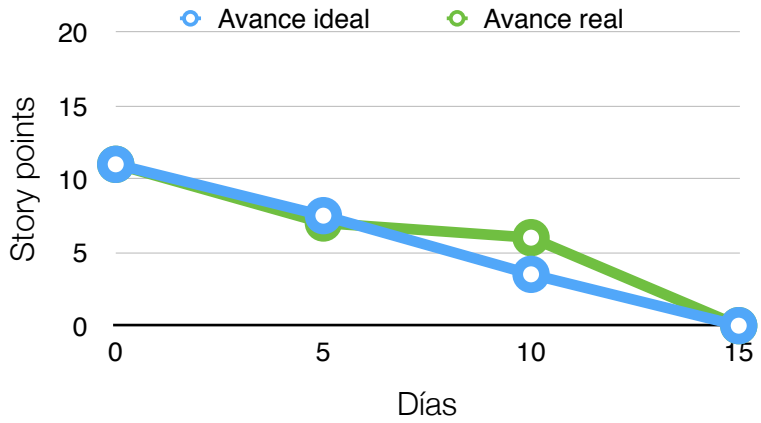

Figura 27: Burndown chart del sprint 2.

4.2.2.6- Retrospectiva

En la siguiente tabla se resumen, a modo de retrospectiva, las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El avance real en el desarrollo ha ido bastante ajustado al teórico.	La implementación total del sistema de seguridad se completó en los últimos días del sprint	No realización de pruebas de seguridad complejas al sistema (p.e.: resistencia a un ataque de tipo man in the middle).

Figura 28: Retrospectiva del sprint 2.

4.2.3- Sprint 3

4.2.3.1- Sprint backlog

En este tercer sprint, que inicia el desarrollo de la segunda release, se continúa con la implementación de funcionalidad en la aplicación Android, desarrollando las historias US6 y US8, orientadas a mostrar al usuario la información disponible en la base de datos sobre sus reservas de tutorías.

Product backlog		
R2	**S3**	**US6, US8**
	S4	US7, US9
R3	F5	
	F6	

Figura 29: Product backlog con las historias de usuario del sprint 3 destacadas.

US6: Consultar un listado de mis reservas de tutorías

Con esta historia, se busca cubrir una de las funcionalidades básicas del sistema: poder consultar en cualquier momento las reservas de tutorías que se tengan activas, con toda su información. Se le ha asignado a esta historia un valor de 5 SP.

US8: Ordenar los listados por fecha u orden de reserva

De cara a desarrollar una aplicación de un uso más cómodo para el usuario, se cuenta con esta historia de usuario, destinada a implementar opciones de ordenación de todos los listados con los que contará la aplicación. Se le ha asignado a esta historia un valor de 4 SP.

Como resultado de este sprint, se espera obtener una versión de la aplicación Android con una vista dedicada al listado de las reservas de tutorías activas, que muestre toda la información asociada a las mismas. Esta sección será accesible desde el menú lateral de navegación de la aplicación, que también se verá mas desarrollado en este sprint.

A nivel backend, se espera que la API soporte las llamadas necesarias para devolver a la aplicación móvil la información de todas las reservas de tutorías, del usuario que las solicite.

4.2.3.2- Detalle del sprint

En este sprint se continúa con el desarrollo de funcionalidad del sistema, realizando avances tanto a nivel backend como a nivel de la aplicación Android o frontend. Particularmente, los avances van orientados al listado de todas las tutorías reservadas almacenadas en la base de datos del sistema.

Backend

Puesto que todo avance en la aplicación cliente Android está prácticamente supeditado en su totalidad a los avances que se realicen en el backend (que es quien aporta toda la información con la que trabaja la aplicación Android), los primeros avances de este sprint se llevaron a cabo sobre la API, implementando una serie de llamadas o peticiones, destinadas a conformar el **contrato de API** (listado con todas las llamadas que soporta la API).

En este sprint se ha trabajado en las siguientes peticiones:

Contrato de API (I)		
Método HTTP	**Ruta**	**Descripción**
GET	/api/user/info	Devuelve la información básica de cada usuario
GET	/api/user/courses	Devuelve la información de los cursos del usuario (nombre y profesores asociados).
GET	/api/user/reserves	Devuelve el listado de reservas de tutorías de ese usuario, con toda la información de cada tutoría.

Figura 30: Peticiones desarrolladas durante el sprint 3.

A la par que se desarrollaban las peticiones, se estructuró la API en grupos de rutas, según el carácter de cada petición. Al final del sprint, la API estaba dividida en:

- Grupo “/api”: grupo que contiene todos los demás grupos de peticiones, y que constituye la base de todas las rutas. Sobre este grupo se aplica la función middleware *requiresLogin*, que comprueba si el cliente que realiza la petición está debidamente autenticado.
- Grupo “/user”: grupo con las peticiones referentes a información sobre el usuario. Todas las peticiones desarrolladas en este sprint se encuentran categorizadas bajo este grupo.

Por último, y también a la par que los avances en el desarrollo del backend, se fue reestructurando la API, separando en módulos diferentes las peticiones mapeadas por el framework Slim de las funciones internas de obtención de información de la base de datos. También se incluyeron módulos auxiliares con funciones de conveniencia, utilizados por los módulos principales.

Aplicación Android (Frontend)

A nivel de la aplicación Android, y haciendo uso de la petición del backend que devuelve la información de todas las reservas relacionadas con el usuario, se diseña una lista que muestre al usuario dicha información. Las celdas de la lista contienen toda la información de cada una de las reservas, mostrando inicialmente una versión reducida de los datos, para ganar en limpieza en la interfaz, y dejando a opción del usuario mostrar el resto de la información, mediante la opción “Más información”.

Se utiliza para cada celda de la lista el estilo de interfaz conocido como *Card* [13]. En cada elemento de la lista se muestran los siguientes datos:

- Nombre de la asignatura.
- Nombre y correo del usuario con el que se tiene relación a través de la reserva: en caso de haber iniciado sesión un alumno, podrá ver los datos del profesor con el que ha reservado. En caso de ser un profesor, podrá ver los datos del alumno que ha reservado la tutoría.
- Tipo de tutoría.
- Motivo de la tutoría.
- Fecha y hora de la tutoría.

Adicionalmente, se añade en cada elemento de la lista la opción para eliminar la reserva, si bien esa opción aun no se implementa, por lo que tan solo queda su representación gráfica.

Todos los listados de la aplicación tendrán la posibilidad de ordenar los resultados por fecha y por orden de reserva, funcionalidad que se implementa en este sprint.

Paralelamente a este desarrollo, y de cara a permitir la navegación entre las diferentes vistas que tendrá la aplicación (actualmente solo tiene la desarrollada en este sprint, correspondiente al listado de tutorías reservadas), se avanza en este sprint en el desarrollo del menú lateral de navegación, que hasta ahora solo contenía de forma funcional la opción de cierre de sesión.

A continuación pueden verse los cambios producidos en la aplicación Android tras este sprint:

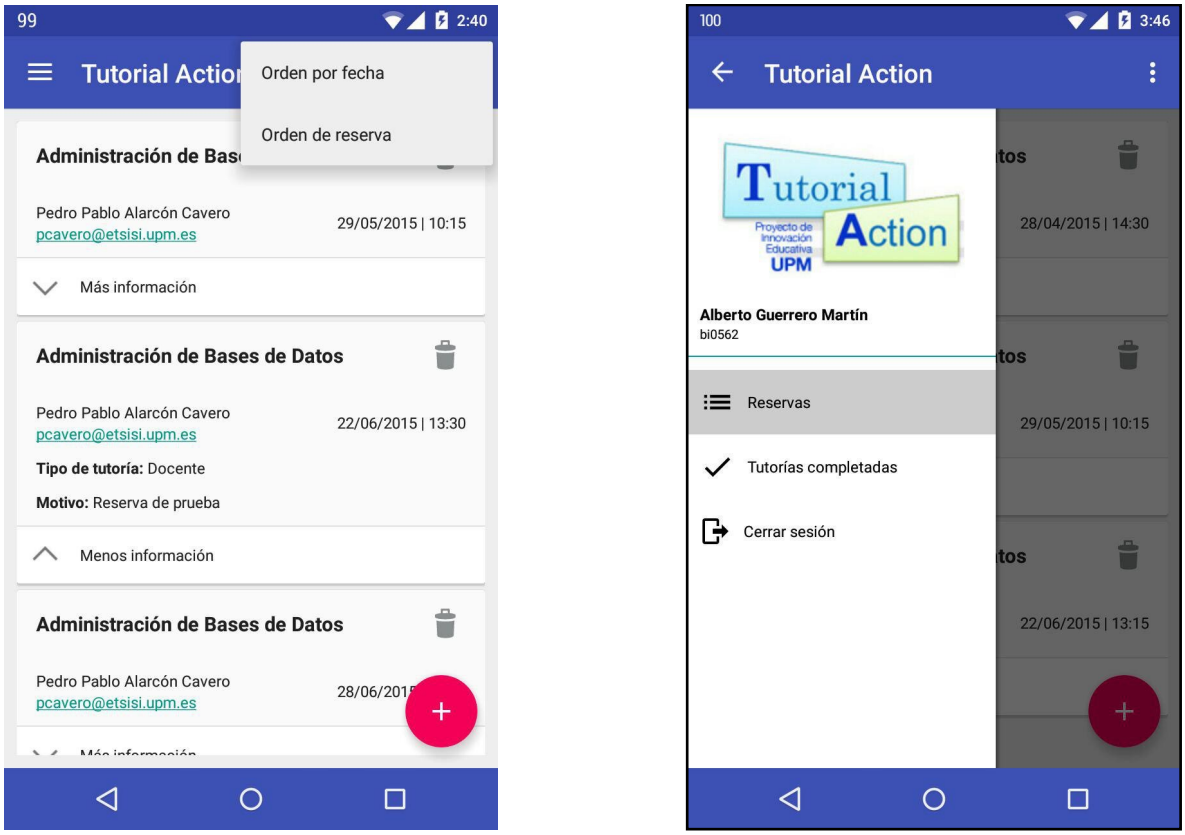

Figura 31: Volcados de la interfaz de usuario.

En la figura 31 pueden verse el listado de reservas de un usuario (izquierda) y la opción de navegación a la vista de reservas en el menú de navegación (derecha).

Control de roles

Si bien la mayor parte del control de roles se lleva a cabo desde el backend, que filtra la información a devolver según el ID del usuario (enviado con cada petición, y mediante el cual obtiene su rol), la aplicación móvil debe controlar el rol del usuario de cara a mostrar o no las opciones de **realizar una nueva reserva** (solo disponible para alumnos) y **añadir una nueva tutoría completada** (solo disponible para profesores).

A lo largo de este sprint, se implementa este control en el listado de reservas, sin llegarse a desarrollar la funcionalidad de añadir una nueva reserva, tarea que queda pospuesta para un futuro sprint. Tal y como puede observarse en las dos imágenes anteriores, en el listado de reservas aparece un botón con la opción “+”, para añadir una nueva reserva. Esto es debido a que el usuario con la sesión iniciada en la aplicación tiene el rol de alumno, rol que le permite realizar reservas de tutorías.

4.2.3.3- Modelo de datos

El modelo de datos no sufre modificación alguna en este sprint:

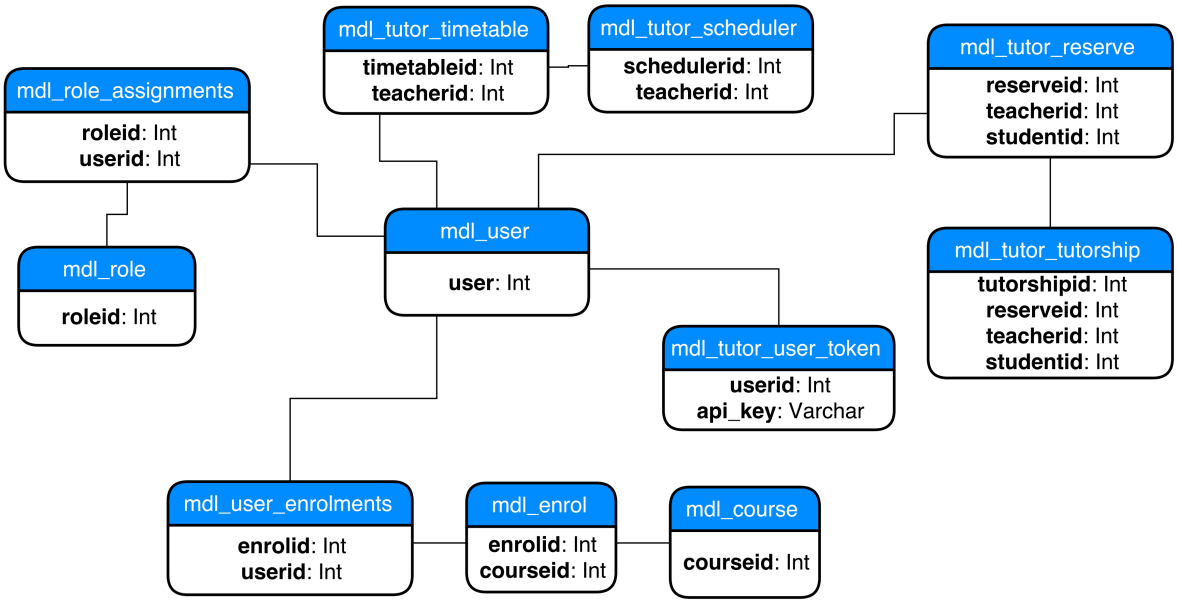

Figura 32: Modelo de datos tras el sprint 3.

4.2.3.4- Diagrama de clases

El diagrama de clases de la API refleja tras este sprint todo el desarrollo llevado a cabo durante el mismo, quedando como sigue:

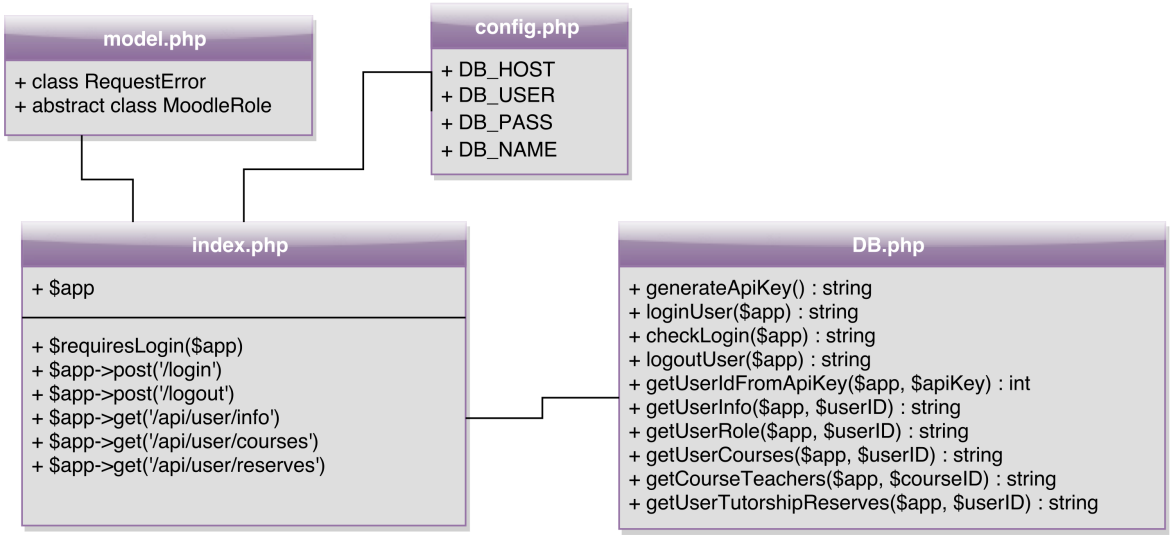

Figura 33: Diagrama de clases de la API tras el sprint 3.

El diagrama de clases de la aplicación Android se amplia con los nuevos módulos añadidos tras los avances de este sprint, quedando como sigue:

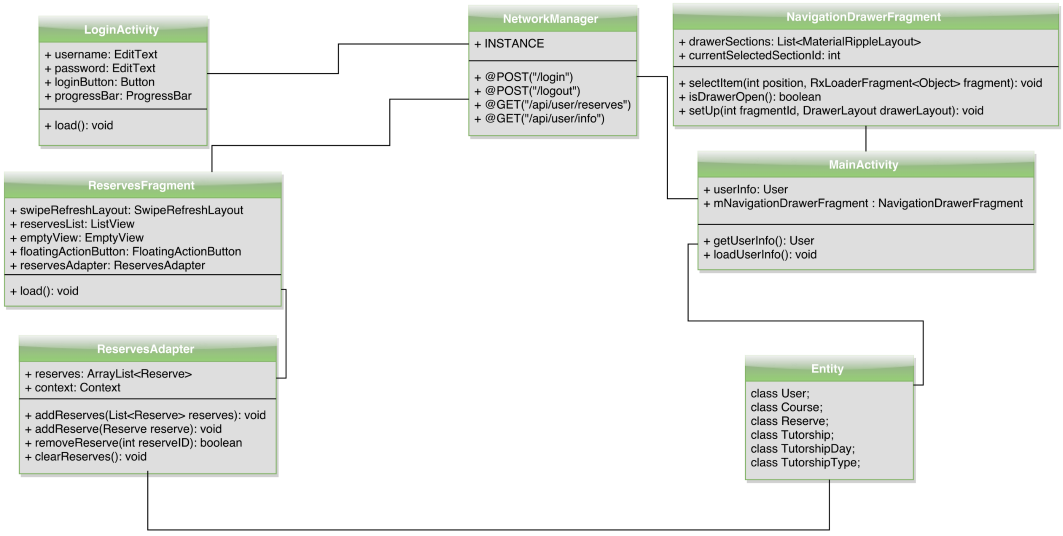

Figura 34: Diagrama de clases de la aplicación Android tras el sprint 3.

4.2.3.5- Sprint review

El resultado de este sprint es una versión de la API con un contrato de API parcialmente implementado, y una aplicación Android con una vista especifica para el listado de reservas de tutorías y con opciones de ordenación del listado, que cubren las historias de usuario US6 y US8.

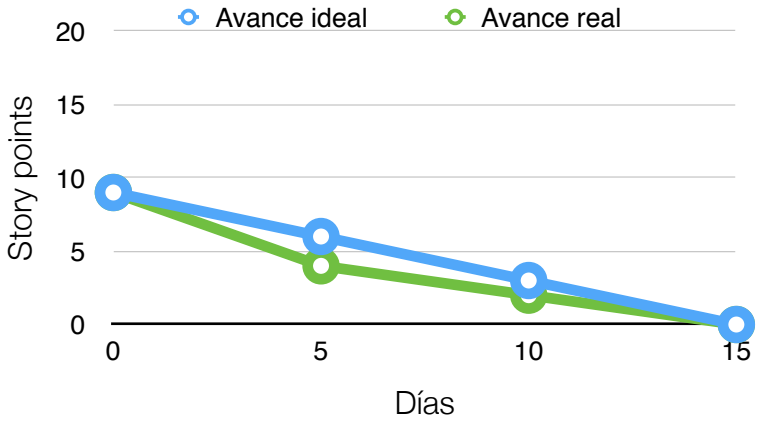

Figura 35: Burndown chart del sprint 3.

4.2.3.6- Retrospectiva

En la siguiente tabla se resumen, a modo de retrospectiva, las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El diseño elegido para la interfaz es acorde con los estándares de diseño para Android. El avance real en el desarrollo ha ido bastante ajustado al teórico.	Infraestimación del sprint: antes de finalizar su duración, estaba completado.	Se implementó funcionalidad en el backend que resultó ser innecesaria.

Figura 36: Retrospectiva del sprint 3.

4.2.4- Sprint 4

4.2.4.1- Sprint backlog

En este cuarto sprint, que cierra el desarrollo de la segunda release, se continúa con la implementación de funcionalidad en la aplicación Android, desarrollando las historias US7 y US9, orientadas a mostrar al usuario la información disponible en la base de datos sobre aquellas tutorías que ya haya completado.
Asimismo, en este sprint se implementa la actualización manual de la información de los listados de tutorías reservadas y tutorías completas.

Product backlog		
R2	S3	US6, US8
	S4	**US7, US9**
R3	F5	
	F6	

Figura 37: Product backlog con las historias de usuario del sprint 4 destacadas.

US7: Consultar un listado de mis tutorías completadas

Con esta historia, se incide en el mismo concepto que presentaba la historia US6, pero en este caso orientado a poder consultar en cualquier momento las tutorías ya completadas por el usuario, con toda su información. Se le ha asignado a esta historia un valor de 5 SP.

US9: Actualización de los listados de reservas y tutorías completadas

Puesto que la base de datos de este sistema no recibe únicamente de entradas de información desde el backend desarrollado en este TFG, sino que desde otras aplicaciones (como el plugin del framework Moodle al que se está portando el sistema) también se pueden realizar reservas de tutorías y registros de tutorías completadas, es vital que la aplicación pueda recargar en cualquier momento la información que recibe de la API. Esta necesidad queda cubierta con esta historia, a la que se le ha asignado un valor de 4 SP.

Como resultado de este sprint, se espera obtener una versión de la aplicación Android con una nueva sección (con su correspondiente opción en el menú lateral) dedicada al listado de las tutorías ya completadas, que muestre toda la información asociada a las mismas.
Asimismo, al finalizar este sprint, el usuario podrá actualizar el contenido de ambos listados de forma manual.

4.2.4.2- Detalle del sprint

En este sprint los avances en las aplicaciones backend y frontend van orientados al listado de todas las tutorías completadas almacenadas en la base de datos del sistema.

Backend

En este sprint se desarrolla la llamada de la API responsable de devolver como respuesta a la aplicación Android toda la información sobre las tutorías ya completadas del usuario.

Esta nueva llamada se añade al contrato de la API, que queda como sigue:

Contrato de API (I)		
Método HTTP	**Ruta**	**Descripción**
GET	/api/user/info	Devuelve la información básica de cada usuario
GET	/api/user/courses	Devuelve la información de los cursos del usuario (nombre y profesores asociados).
GET	/api/user/reserves	Devuelve el listado de reservas de tutorías de ese usuario, con toda la información de cada tutoría.
GET	**/api/user/completed**	**Devuelve el listado de tutorías completas de ese usuario.**

Figura 38: Contrato de API tras el sprint 4.

Aplicación Android (Frontend)

Haciendo uso de la nueva petición del backend que devuelve la información de todas las tutorías ya completadas por el usuario, se desarrolla una lista similar a la de reservas, que muestre al usuario un listado de sus tutorías completadas.

El diseño es similar al del listado de reservas, se utiliza para cada celda el estilo *Card*. En cada elemento de la lista se muestran los siguientes datos:

- Nombre de la asignatura.
- Nombre y correo del usuario con el que se tiene relación a través de la reserva: en caso de haber iniciado sesión un alumno, podrá ver los datos del profesor con el que ha reservado. En caso de ser un profesor, podrá ver los datos del alumno que ha reservado la tutoría.
- Tipo de tutoría.
- Motivo de la tutoría.
- Fecha y hora de la tutoría.

- Duración de la tutoría: este campo supone una novedad con respecto al listado de tutorías reservadas.

Tal y como se implementó en el anterior sprint, este listado tiene la posibilidad de ordenar los resultados por fecha y por orden de reserva. Asimismo, tanto este listado como el correspondiente a las tutorías reservadas pueden actualizarse manualmente.

Como ya se hiciese en el sprint anterior, al haberse desarrollado una nueva vista de la aplicación, se añade la opción de navegación a dicha vista en el menú lateral de la aplicación.
A continuación pueden verse los cambios introducidos en la aplicación Android:

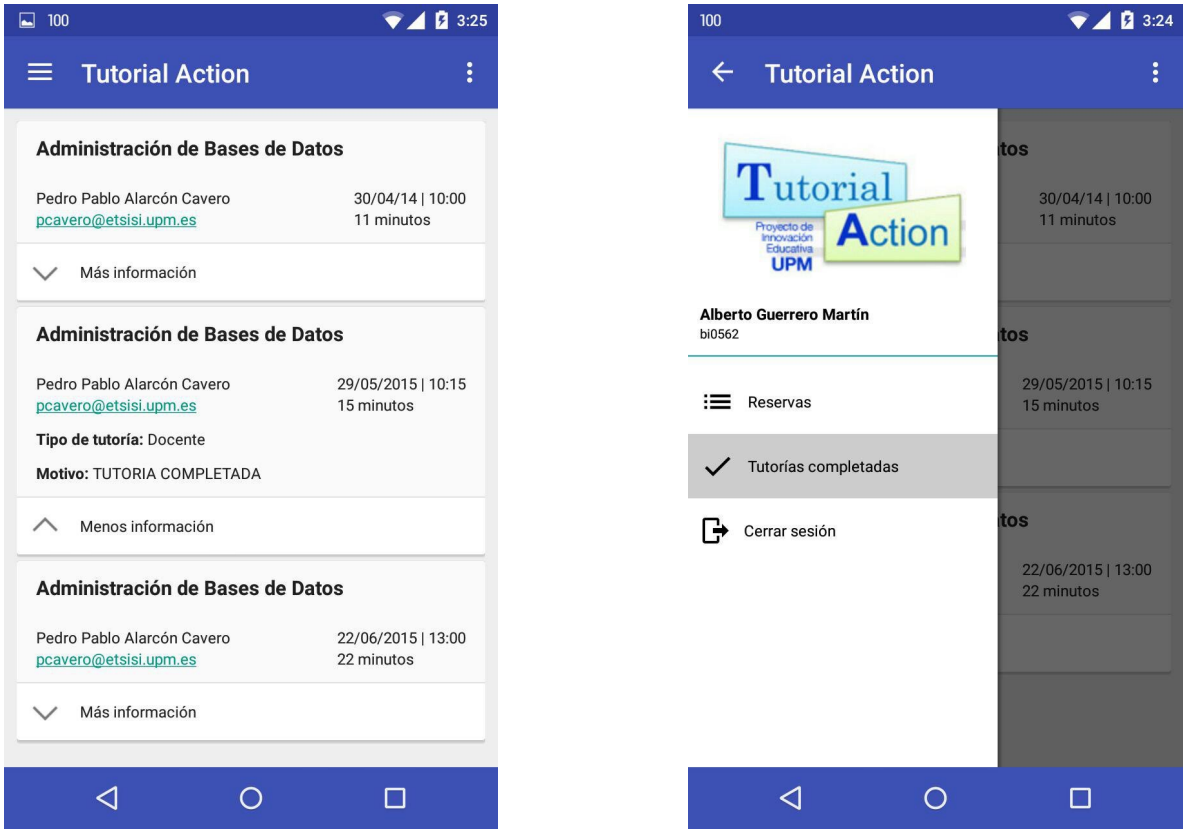

Figura 39: Volcados de la interfaz de usuario.

En la figura 39 pueden verse la vista del listado de tutorías completadas por el usuario (izquierda) y la opción de navegación desde el menú lateral (derecha).
Como ya se comentaba en el sprint anterior, en ambas vistas se ha llevado a cabo el control de roles necesario para mostrar o no el botón con la opción “+”. En las dos imágenes anteriores, el usuario que ha iniciado sesión tiene rol de alumno, por lo que no puede utilizar la opción de registrar nuevas tutorías completadas.

4.2.4.3- Modelo de datos

El modelo de datos no sufre modificación alguna en este sprint:

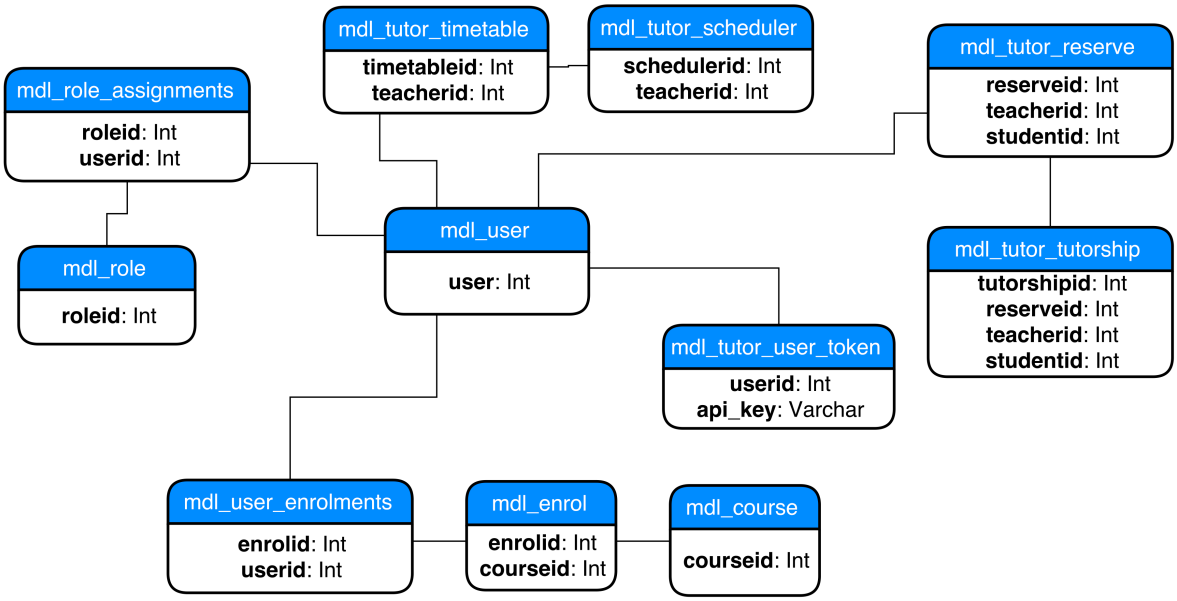

Figura 40: Modelo de datos tras el sprint 4.

4.2.4.4- Diagrama de clases

En este sprint, se añade al diagrama de clases de la API la llamada desarrollada, quedando como sigue:

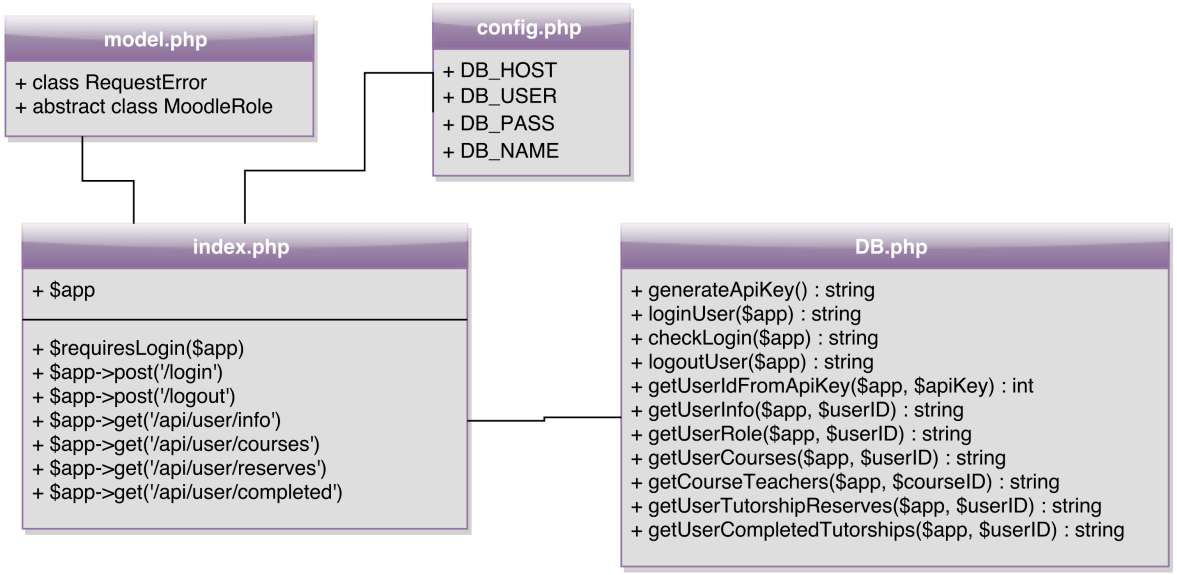

Figura 41: Diagrama de clases de la API tras el sprint 4.

Tras este sprint, el diagrama de clases de la aplicación Android queda como sigue:

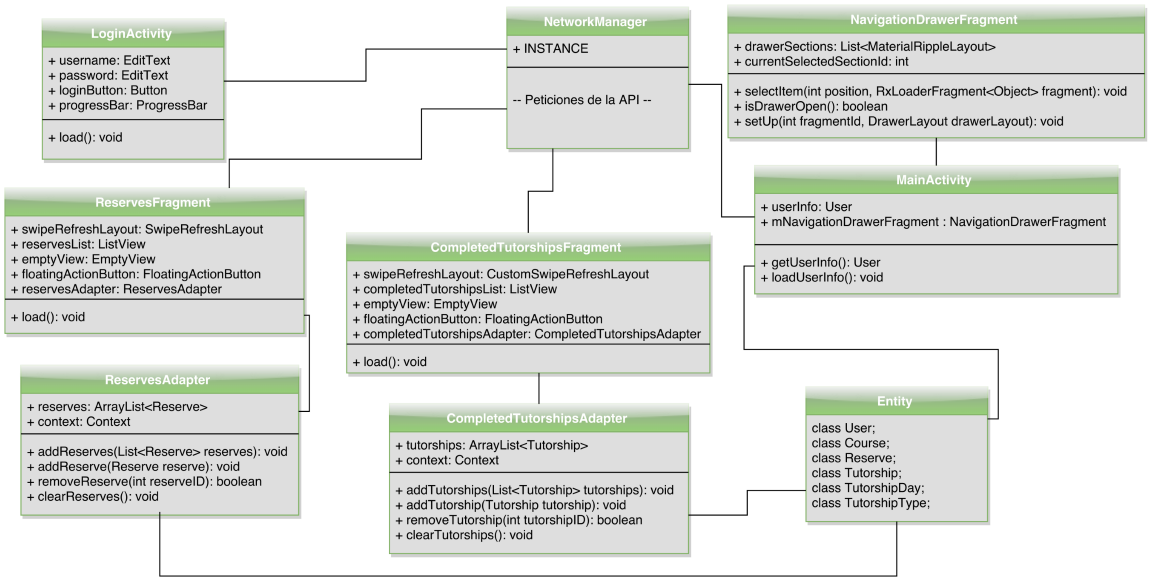

Figura 42: Diagrama de clases de la aplicación Android tras el sprint 4.

4.2.4.5- Sprint review

El resultado de este sprint es una versión de la API con el añadido de la llamada encargada de devolver la información de las tutorías completadas, y una aplicación Android con una vista especifica para el listado de tutorías completadas, que es actualizable. Se cubren las historias de usuario US6 y US8.

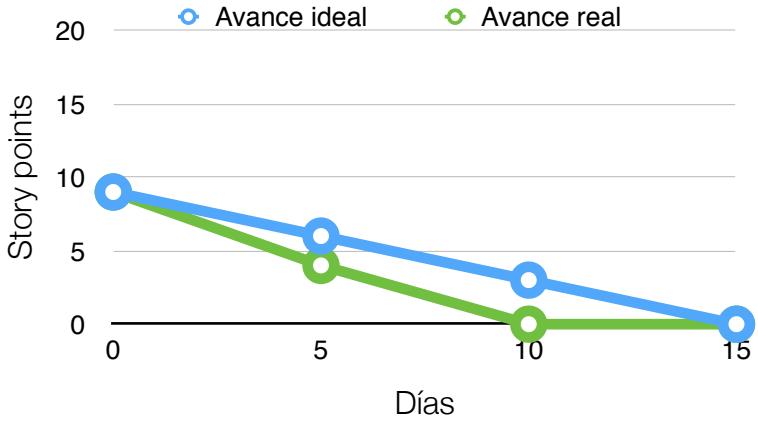

Figura 43: Burndown chart del sprint 4.

4.2.4.6- Retrospectiva

En la siguiente tabla se resumen las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El diseño elegido para la interfaz es acorde con los estándares de diseño para Android. El avance real en el desarrollo ha ido bastante ajustado al teórico.	Infraestimación del sprint: antes de finalizar su duración, estaba completado.	

Figura 44: Retrospectiva del sprint 4.

4.2.5- Sprint 5

4.2.5.1- Sprint backlog

En este quinto sprint, que comienza el desarrollo de la tercera y última release del proceso de desarrollo, se implementa una de las funcionalidades clave y objetivo del proyecto: la reserva de tutorías desde la aplicación móvil, cubriendo la historia de usuario US10. A la par, se desarrolla la funcionalidad correspondiente a la historia de usuario US12, consulta de las horas disponibles de un profesor para tutorías, funcionalidad necesaria para realizar la reserva de tutorías.
Tras este sprint, quedará implementada por completo en la aplicación Android la gestión de las reservas de tutorías.

Product backlog		
R3	**S5**	**US10, US12**
	S6	US11, US13

Figura 45: Product backlog con las historias de usuario del sprint 5 destacadas.

US10: Reservar una tutoría

La funcionalidad asociada a esta historia, fundamental para el sentido de todo el proyecto, permitirá a los usuarios realizar reservas de tutorías desde la aplicación móvil. Dada su trascendencia para el proyecto, se le ha asignado a esta historia un valor de 8 SP.

US12: Consultar horas disponibles para reserva una tutoría

Para realizar una reserva de tutoría, es necesario indicar, y por tanto conocer, el día y hora disponibles del profesor con el que se reserva la tutoría. Con esta historia, a la que se le ha asignado un valor de 4 SP, se da soporte a dicha funcionalidad.

Como resultado de este sprint, se espera obtener una versión de la aplicación Android en la que sea posible gestionar al 100% las reservas de tutorías.

4.2.5.2- Detalle del sprint

En este sprint los avances en las aplicaciones backend y frontend van orientados a la reserva de tutorías y la consulta de la disponibilidad de un profesor.

Backend

En este sprint se prosigue con el desarrollo de la API, implementando en este caso las llamadas o peticiones necesarias para la reserva de una tutoría desde la aplicación Android.

El contrato de la API, tras ser actualizado, queda de la siguiente manera:

Contrato de API (I)		
Método HTTP	**Ruta**	**Descripción**
GET	/api/user/info	Devuelve la información básica de cada usuario
GET	/api/user/courses	Devuelve la información de los cursos del usuario (nombre y profesores asociados).
GET	/api/user/reserves	Devuelve el listado de reservas de tutorías de ese usuario, con toda la información de cada tutoría.
GET	/api/user/completed	Devuelve el listado de tutorías completas de ese usuario.
GET	**/api/user/:teacherID/ timetable**	**Devuelve las franjas horarias libres para tutorías del profesor solicitado (teacherID).**
POST	**/api/reserves/create**	**Registra una nueva reserva de tutoría en la base de datos.**

Figura 46: Contrato de API tras el sprint 5.

Con estas nuevas peticiones, se añade un nuevo grupo de peticiones para estructurar mejor la API:

- Grupo “/reserves”: grupo con las peticiones referentes a las reservas de tutorías.

Aplicación Android (Frontend)

Tras añadir y probar en el backend las peticiones destacadas en la tabla superior, se desarrolla la vista correspondiente a la reserva de una tutoría.

La vista que permite registrar una nueva reserva de tutoría (vista a la que solo podrán acceder aquellos usuarios con rol de alumno) consta de una serie de campos a completar, que se corresponden con los datos necesarios para registrar la reserva.

La interfaz desarrollada queda de la siguiente manera:

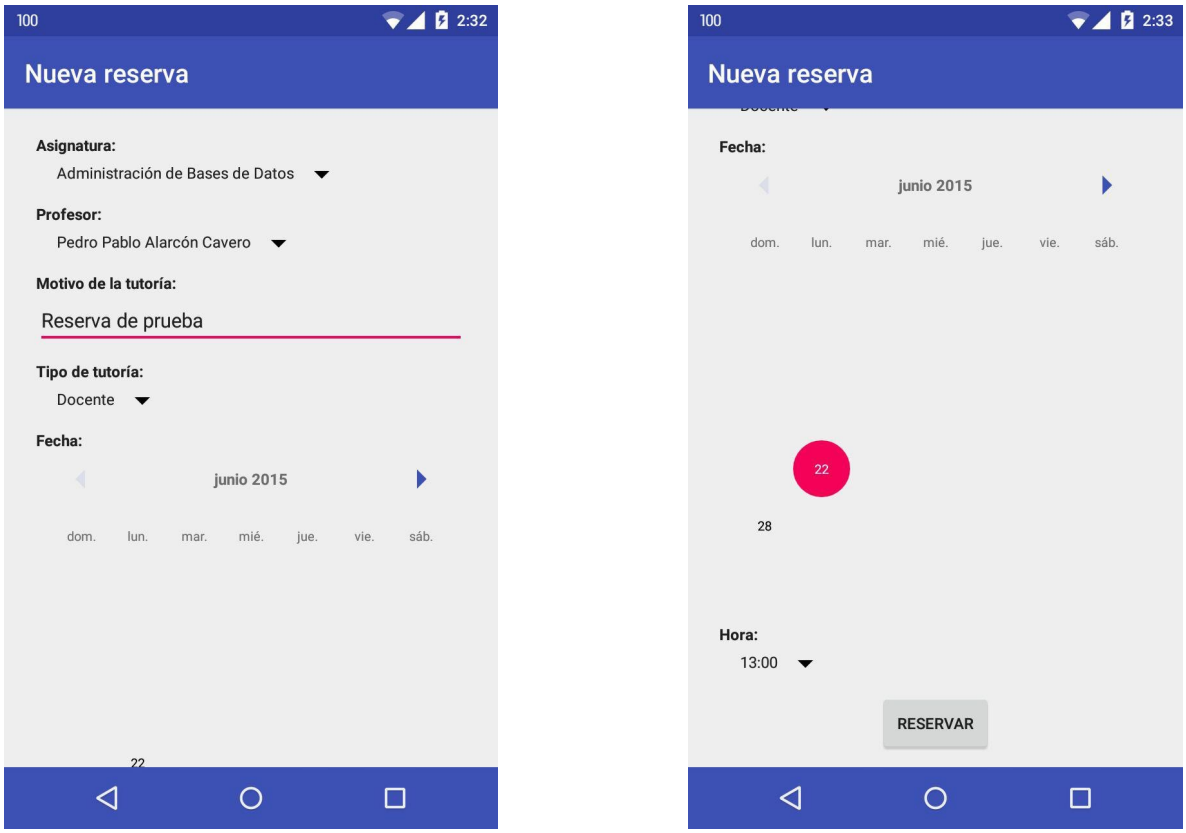

Figura 47: Vista de registro de una nueva reserva de tutoría.

Cabe destacar en esta vista el calendario para seleccionar el día de la reserva, que tan solo muestra los días disponibles del profesor, y no permite ver días anteriores al actual. Para lograr esta funcionalidad, fue necesario utilizar una librería *open-source* desarrollada por un tercero, Material-CalendarView [14], que además tuvo que ser modificada, ya que inicialmente no ofrecía la opción de bloquear la selección de días concretos.

4.2.5.3- Modelo de datos

El modelo de datos no sufre modificación alguna en este sprint:

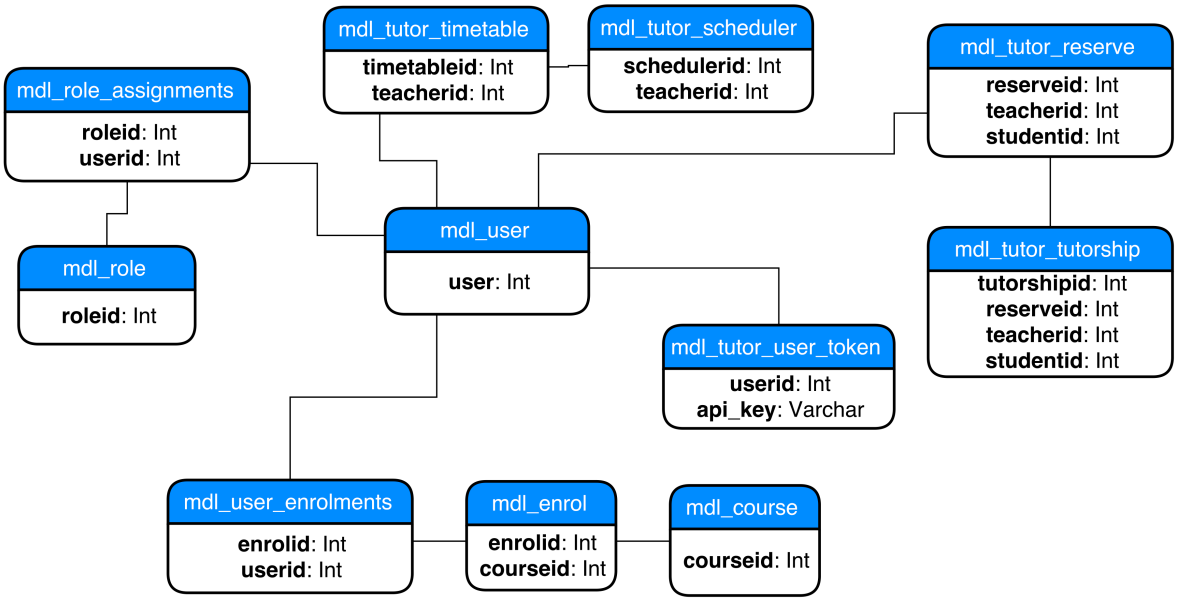

Figura 48: Modelo de datos tras el sprint 5.

4.2.5.4- Diagrama de clases

Tras este sprint, se añade al diagrama de clases de la API las llamadas desarrolladas, quedando como sigue:

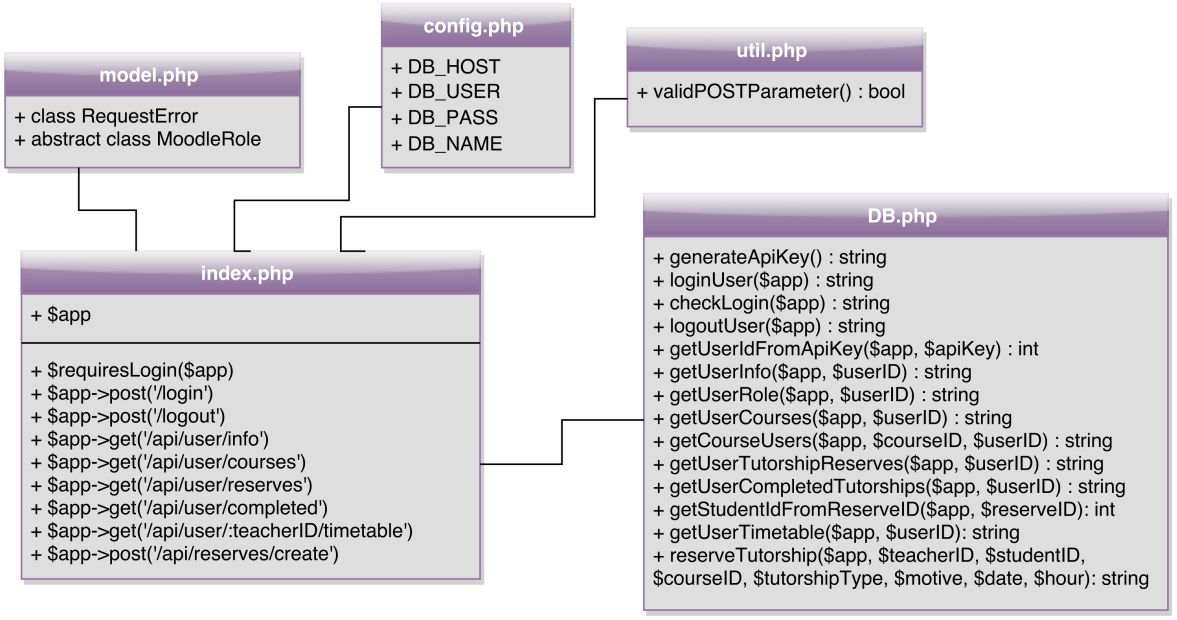

Figura 49: Diagrama de clases de la API tras el sprint 5.

Tras este sprint, el diagrama de clases de la aplicación Android queda como sigue:

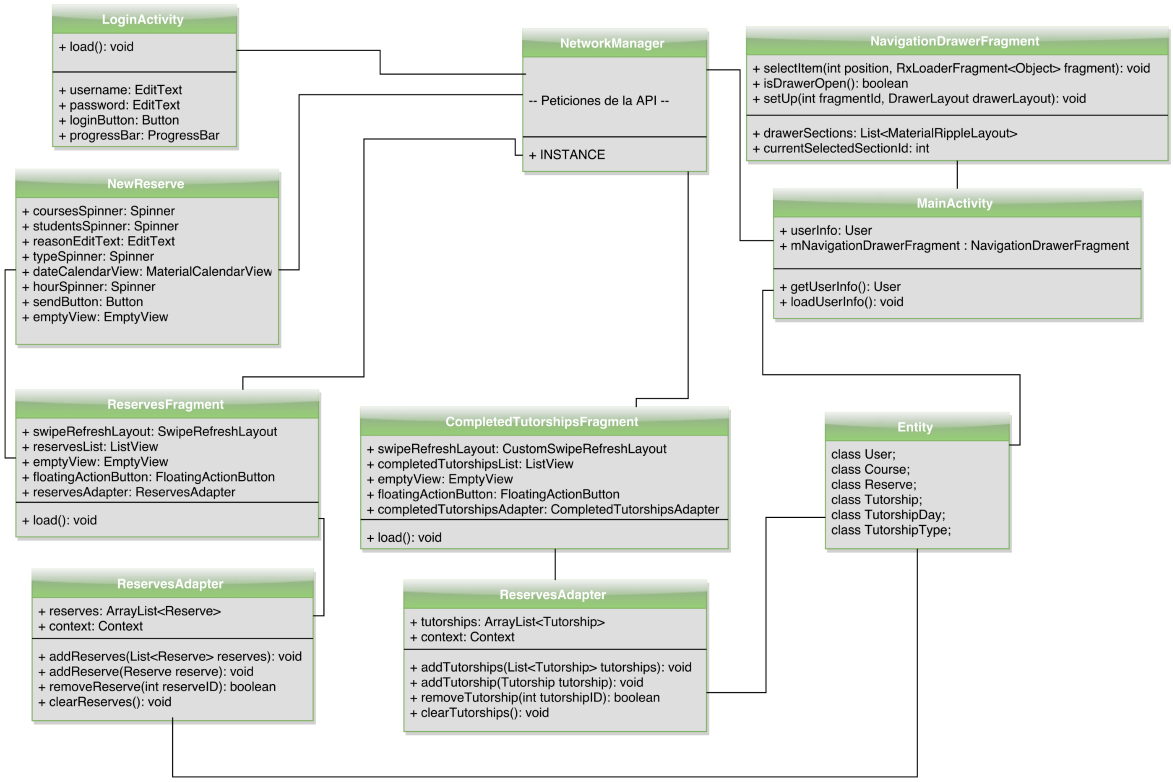

Figura 50: Diagrama de clases de la aplicación Android tras el sprint 5.

4.2.5.5- Sprint review

El resultado de este sprint es una versión de la API con las llamada necesarias para realizar una reserva de una tutoría, y una aplicación Android con una vista especifica para realizar dicha reserva, en la que además se puede consultar, mediante un calendario, las horas disponibles de cada profesor. Por tanto, tras este sprint quedan cubiertas las historias de usuario US10 y US12.

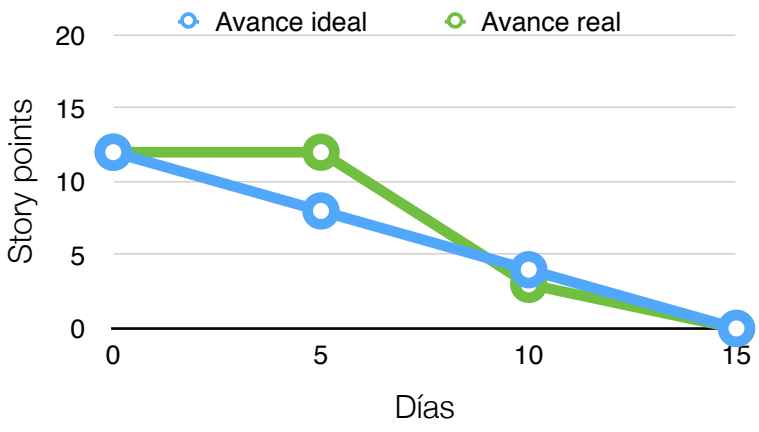

Figura 51: Burndown chart del sprint 5.

4.2.5.6- Retrospectiva

En la siguiente tabla se resumen, a modo de retrospectiva, las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El diseño elegido para la interfaz es acorde con los estándares de diseño para Android. El avance real en el desarrollo ha ido bastante ajustado al teórico. Modificación la librería del calendario, ya que permite consultar de forma muy cómoda las horas libres de cada profesor para reservar la tutoría.	Sobrecarga del sprint con dos funcionalidades complejas.	

Figura 52: Retrospectiva del sprint 5.

4.2.6- Sprint 6

4.2.6.1- Sprint backlog

En el sprint final del proceso de desarrollo, con el que se cierra la tercera y última release, se implementarán las últimas funcionalidades, cubiertas por las historias de usuario US11 y US13. Además, se añadirá a la aplicación Android la integración con el sistema de captura de estadísticas Google Analytics [15].

Con este sprint, se dará por finalizado el desarrollo, y se obtendrá una versión funcional del sistema.

Product backlog		
R3	S5	US10, US12
	S6	**US11, US13**

Figura 53: Product backlog con las historias de usuario del sprint 6 destacadas.

US11: Cancelar una reserva de tutoría

Esta historia trata la funcionalidad de la cancelación de reservas de tutorías, opción necesaria dentro del sistema. Se le ha asignado a esta historia un valor de 3 SP.

US13: Registrar una tutoría completada

Esta historia se corresponde con el último paso en el ciclo de vida de cualquier tutoría. Una vez completada, el profesor podrá registrar en el sistema esa tutoría como completada, tanto si previamente fue reservada como si no. Se le ha asignado a esta historia un valor de 5 SP.

Como resultado de este sprint, se espera obtener la versión funcional 1.0 del sistema completo.

4.2.6.2- Detalle del sprint

En este sprint los avances en las aplicaciones backend y frontend se enfocan en el registro de una tutoría ya completada, aunque también se desarrolla la cancelación de reservas y la monitorización del sistema mediante Google Analytics.

Backend

En este sprint se finaliza el desarrollo de la API con las llamadas correspondientes al registro de tutorías completadas y a la cancelación de las reservas.
El contrato de la API, tras ser actualizado, queda de la siguiente manera:

Contrato de API (I)		
Método HTTP	**Ruta**	**Descripción**
GET	/api/user/info	Devuelve la información básica de cada usuario
GET	/api/user/courses	Devuelve la información de los cursos del usuario (nombre y profesores asociados).
GET	/api/user/reserves	Devuelve el listado de reservas de tutorías de ese usuario, con toda la información de cada tutoría.
GET	/api/user/completed	Devuelve el listado de tutorías completas de ese usuario.
GET	/api/user/:teacherID/ timetable	Devuelve las franjas horarias libres para tutorías del profesor solicitado (teacherID).
POST	/api/reserves/create	Registra una nueva reserva de tutoría en la base de datos.
POST	**/api/reserves/ complete/:reserveID**	**Completa una tutoría previamente reservada.**
DELETE	**/api/reserves/ remove/:reserveID**	**Elimina de la base de datos la reserva solicitada (reserveID).**
POST	**/api/tutorships/ create**	**Registra una nueva tutoría completada en la base de datos.**

Figura 54: Contrato de API tras el sprint 6.

Con estas nuevas peticiones, se añade un nuevo grupo de peticiones para estructurar mejor la API:

- Grupo "/tutorships": grupo con las peticiones referentes a las tutorías ya completadas.

Como característica a resaltar, las tutorías completadas pueden registrarse a través de dos vías, para las cuales hay sendas peticiones desarrolladas:

- Si la tutoría había sido reservada, se **completa** la reserva ("/api/reserves/ complete/:reserveID").
- Si la tutoría **no** había sido reservada, simplemente se **registra** ("/api/tutorships/ create").

Aplicación Android (Frontend)

La vista que permite registrar una nueva tutoría completada (a la que solo podrán acceder aquellos usuarios con rol de profesor) consta, al igual que la vista de nueva reserva, con una serie de campos a completar. La interfaz desarrollada queda de la siguiente manera:

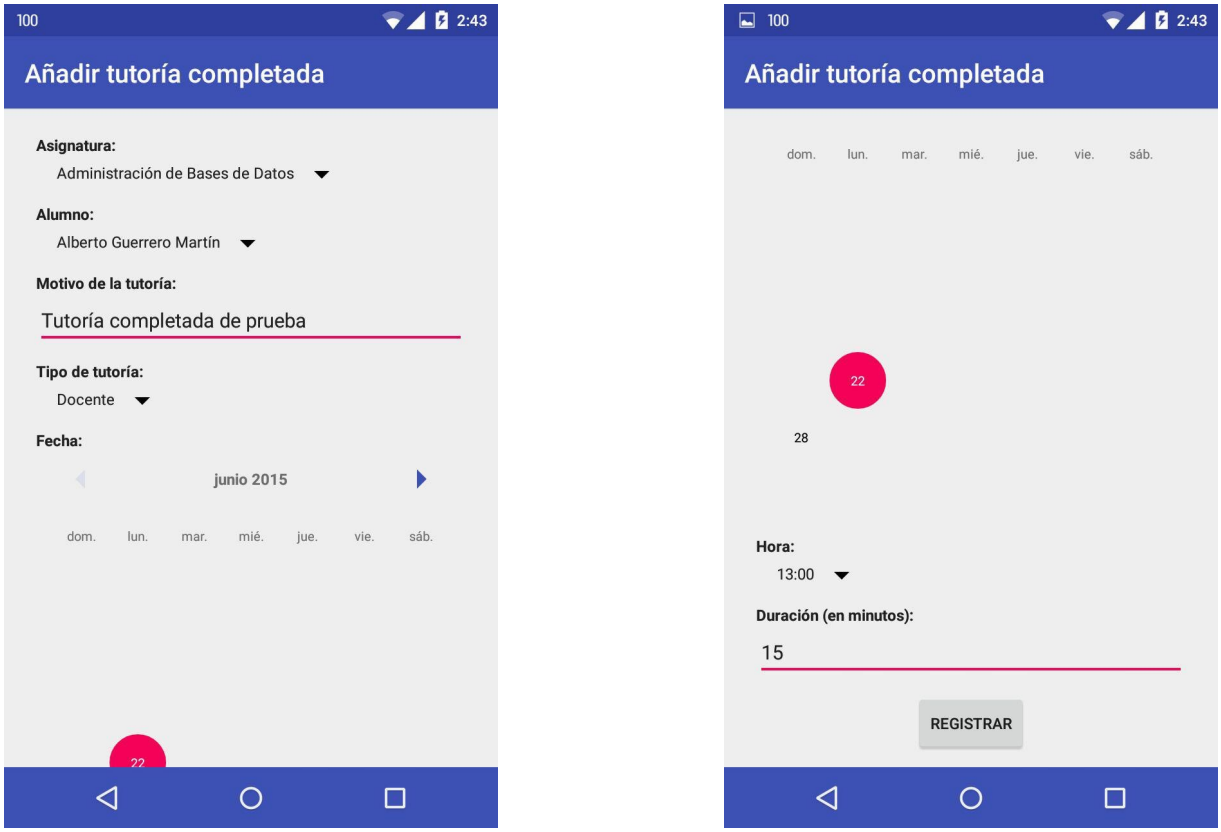

Figura 55: Vista de registro de una nueva tutoría completada.

No obstante, hay una segunda forma de registrar una nueva tutoría completada: a través del listado de reservas, utilizando la opción de completar reserva:

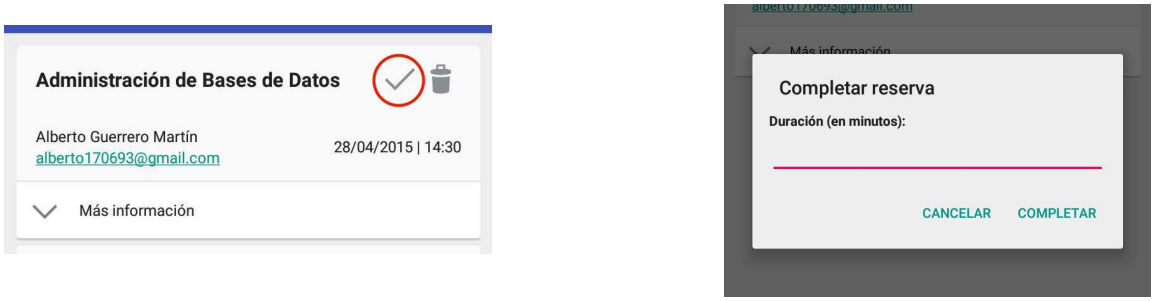

Figura 56: Opción para completar una tutoría reservada.

Esta opción permite completar de forma rápida las tutorías que hubiesen sido previamente reservadas, necesitando únicamente la duración de la tutoría como información adicional.

En cuanto a la opción de eliminar reserva, pasa a ser completamente funcional.

Google Analytics

Una vez este sistema se encuentre en producción, y disponible para su uso, será de gran interés poder conocer ciertas estadísticas como por ejemplo su volumen de uso, la opción mas utilizada por los usuarios, y similares.

Ante esta situación, el product owner y el desarrollador coincidieron en que la aplicación Android debía incluir monitorización, utilizando la herramienta Google Analytics. En este sprint final, se realizó la implementación correspondiente a la monitorización.

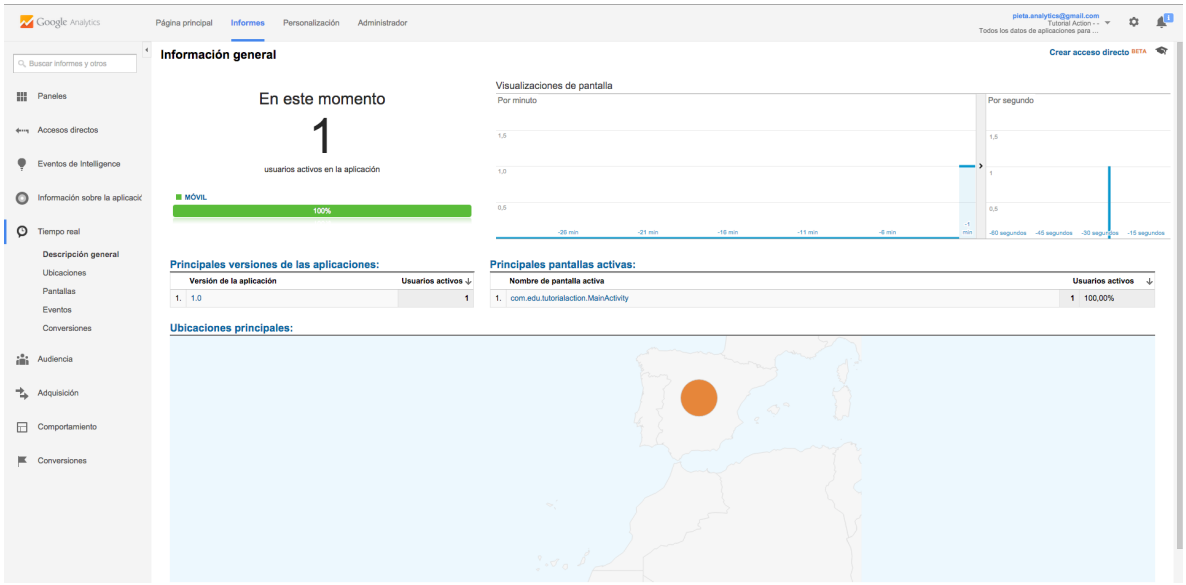

Figura 57: Panel de información de Google Analytics.

4.2.6.3- Modelo de datos

Al no haberse modificado en este sprint, la versión final del modelo de datos es la siguiente:

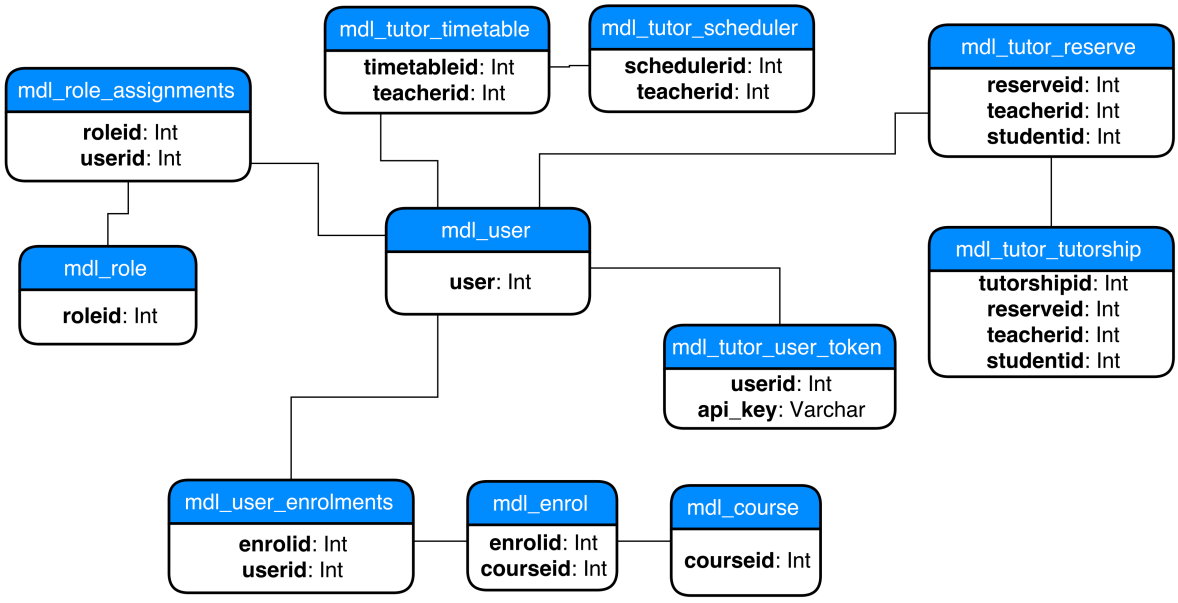

Figura 58: Modelo de datos tras el sprint 6.

4.2.6.4- Diagrama de clases

Tras desarrollar la funcionalidad correspondiente a este sprint, los diagramas de clases definitivos de la versión 1.0 del sistema, tanto para la API como para la aplicación Android, son los siguientes:

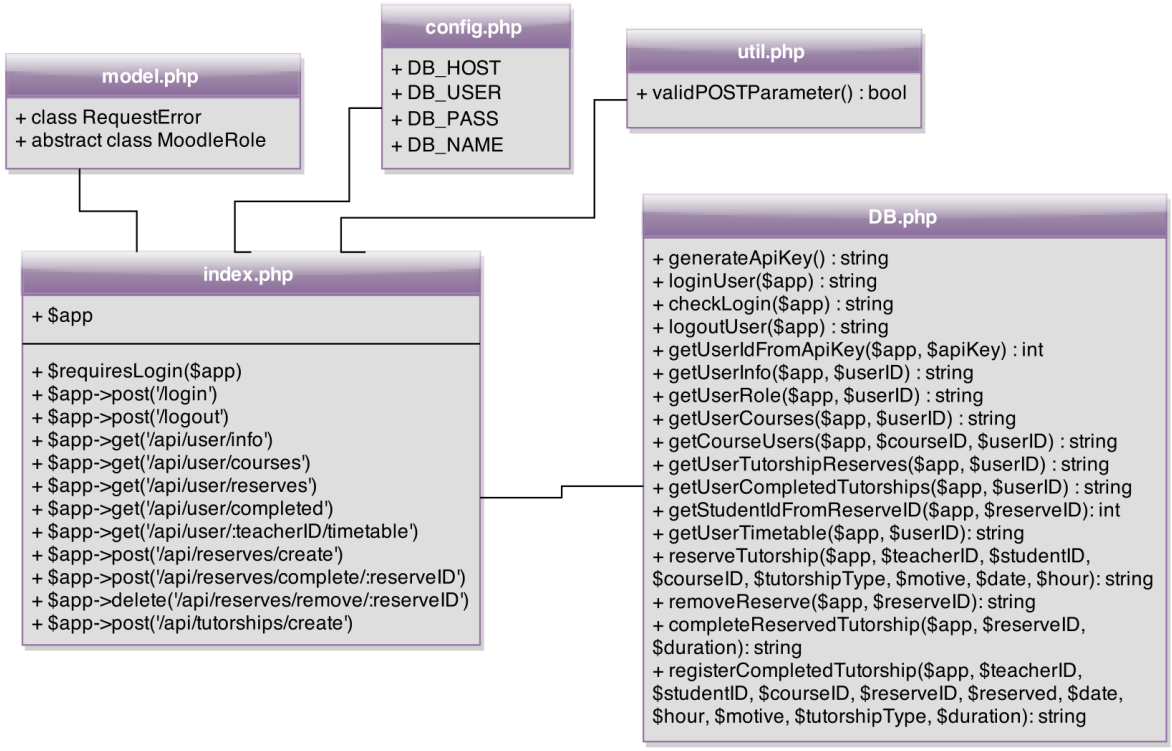

Figura 59: Diagrama de clases de la API tras el sprint 6.

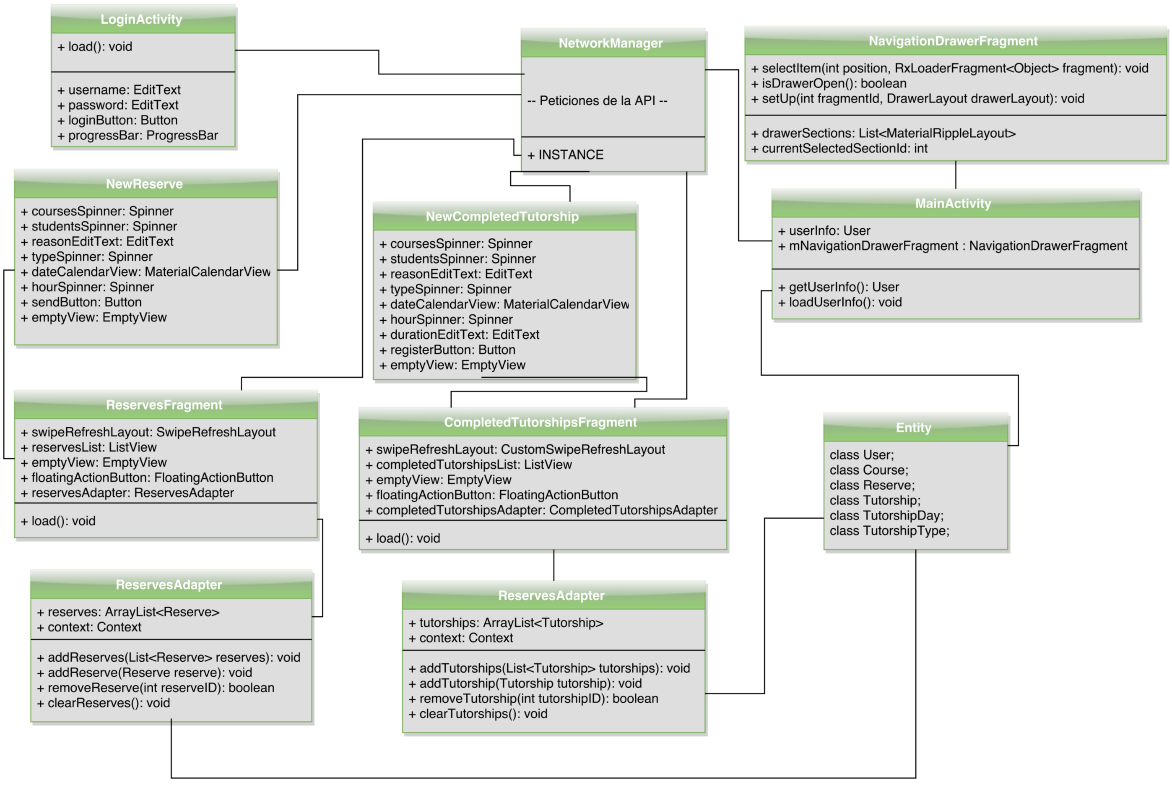

Figura 60: Diagrama de clases de la aplicación Android tras el sprint 6.

4.2.6.5- Sprint review

El resultado de este sprint, tras quedar cubiertas las historias de usuario US11 y US13, es la versión 1.0 del sistema, completamente funcional, y lista para ser pasada a producción y distribuida entre los usuarios.

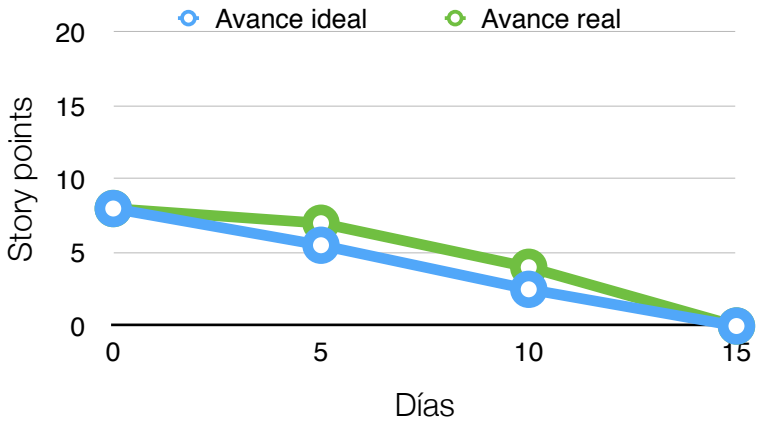

Figura 61: Burndown chart del sprint 5.

4.2.6.6- Retrospectiva

En la siguiente tabla se resumen, a modo de retrospectiva, las conclusiones finales de este sprint:

OK	▲ (mejorable)	KO
El diseño elegido para la interfaz es acorde con los estándares de diseño para Android. El avance real en el desarrollo ha ido bastante ajustado al teórico. Se completa este último sprint antes de la fecha deadline del proyecto.	A lo largo de todo el sprint, se arrastra un ligero retraso en el desarrollo, aunque finalmente se completa el sprint a tiempo.	

Figura 62: Retrospectiva del sprint 5.

5- Conclusiones y trabajo futuro

5.1- Conclusiones

Como conclusión principal de este TFG, destaco que se han conseguido todos los objetivos propuestos al inicio del mismo. Se ha obtenido como resultado de este trabajo un sistema de gestión de tutorías a través de una aplicación móvil, completamente funcional y listo para lanzarse a los usuarios en su versión de producción.

En cuanto a los objetivos específicos de este TFG:

- Se han llevado a cabo **mejoras en el modelo de datos** del que constaba el sistema al inicio del trabajo, enfocados a la implantación del sistema de seguridad.
- Se ha llevado a cabo el **desarrollo de una API** funcional, que trabaja junto al modelo de datos mencionado, y provee de funcionalidad a las aplicaciones.
- Se ha desarrollado una **aplicación cliente para el sistema operativo móvil Android**, completamente funcional y lista para que el conjunto de usuarios objetivo, alumnos y profesores, haga uso de ella para gestionar sus tutorías.
- Como objetivo interno de cara al proyecto PIETA, se ha dotado a la aplicación móvil de un **sistema de captura de estadísticas de uso** de la misma, a través de la plataforma Google Analytics, que será de gran utilidad para en el futuro extraer conclusiones sobre la mejora que está suponiendo el sistema desarrollado para los usuarios, así como para proponer posibles mejoras y/o líneas de trabajo.

Como conclusión particular, quiero destacar la enorme mejora que este trabajo ha supuesto en mis conocimientos y experiencia en el uso de las metodologías ágiles de desarrollo. Desde el aprendizaje y uso del Inception Deck, herramienta con la que no había trabajado con anterioridad, hasta el sistema de sprints, este trabajo me ha permitido comprobar las ventajas del uso de estas metodologías frente a las mas tradicionales, especialmente en los tiempos de desarrollo y en la calidad final del producto.

Para finalizar, me dirijo a cualquier lector que este documento pueda tener, deseándole que encuentre esta memoria como un buen ejemplo del uso de las metodologías ágiles de desarrollo, así como un buen sistema de uso y gestión de las tutorías, que tan necesarias son para los alumnos en nuestra vida universitaria.

5.2- Trabajo futuro

A la vista del producto finalmente generado por este TFG, se proponen las siguientes líneas de trabajo y mejora:

- **Desarrollar aplicaciones para otras plataformas móviles**: por su volumen de usuarios, el desarrollo de una aplicación para la plataforma iOS equivalente a la desarrollada en este TFG para la plataforma Android supondría un gran paso en el ejercicio de captación de usuarios. También convendría estudiar desarrollos para otras posibles plataformas (Windows Phone, Firefox OS, etc.).
- **Aumentar la funcionalidad**: opciones como la edición de reservas ya realizadas, la modificación de los horarios de un profesor desde las apps móviles, o un sistema completo de notificaciones automáticas son algunas de las posibles vías de mejora del sistema actual.
- **Potenciar la captura de estadísticas**: el sistema desarrollado hace un uso simple de captura de estadísticas. Una posible vía de trabajo sería aumentar la cantidad y calidad de las estadísticas recogidas, enfocándolas a tareas útiles tanto para el nivel de negocio como el de desarrollo (p.e. generación automática de informes de uso).

6- Referencias

[1]: Grupo de Innovación Educativa Tutorial Action (GIETA): http://gieta.eui.upm.es/

[2]: Proyecto de Innovación Educativa Tutorial Action (PIETA): http://pieta.eui.upm.es/web/presentacion.html

[3]: Artículo REDU: http://red-u.net/redu/index.php/REDU/article/view/882

[4]: Aplicación web Tutorial Action: http://pieta.eui.upm.es/

[5]: Estado actual de la metodología Scrum: http://www.scrumprimer.org/

[6]: *"The Agile Samurai"*, Jonathan Rasmusson: https://pragprog.com/book/jtrap/the-agile-samurai

[7]: Artículo de Agility Software "Scrum Whys: Fixed-Length Sprints?": http://www.agilitysw.com/blog/31-scrum-whys-fixed-length-sprints

[8]: PHP: http://php.net/

[9]: Slim Framework: http://www.slimframework.com/

[10]: PhpStorm IDE: https://www.jetbrains.com/phpstorm/

[11]: JetBrains: https://www.jetbrains.com/

[12]: Protocolo SSL: http://es.wikipedia.org/wiki/Transport_Layer_Security

[13]: Android Card View: https://developer.android.com/training/material/lists-cards.html#CardView

[14]: Material Calendar View: https://github.com/prolificinteractive/material-calendarview

[15]: Plataforma Google Analytics: http://www.google.es/intl/es/analytics/

Todas las referencias han sido visitadas y comprobadas a fecha 16/06/2015.